N. 26 - NOVEMBRE 1990
Sped. in abb. post. gr. III/70
L. 15.000
68000
AMIGA BYTE
by Elettronica 2000
SUL DISCO
PONGO 50 LIVELLI DI STRATEGIA
MEGAWB UN GRANDISSIMO WORKBENCH
TITANCRUNCH CRUNCHER SALVAMEMORIA
SMARTICON MENO FINESTRE, PIÙ MEMORIA
VSNAP TAGLIA/INCOLLA TESTO E GRAFICA
WBSHADOW WORKBENCH IN 3D
IFF2EXE IMMAGINI AUTODISPLAY
ANIMPOINTERS NUOVI PUNTATORI ANIMATI
TIPS & TRICKS
Linguaggi
HISOFT BASIC
Musica
TFMX, DELTA MUSIC, FUTURE COMPOSER A CONFRONTO
I GIOCHI NOVITÀ
Ray Tracing
EFFETTI E ANIMAZIONI CON TURBOSILVER
Megagame
DEJA VU II ISLAND OF LOST HOPE
Didattica
FAI DA TE LA TUA STARTUP-SEQUENCE
AMIGA BYTE
Grafica
REAL 3D
Tools
EXTEND 1.3
THE ART DEPARTMENT
T.A.C.L.

N. 26 - NOVEMBRE 1990

Direttore
SIRA ROCCHI

Direzione Editoriale
MARIO MAGRONE

Direzione Tecnica
GIANCARLO CAIRELLA

Segreteria di Redazione
SILVIA MAIER

Grafica
NADIA MARINI

Fotografie
MARIUS LOOK

Copertina
FRANCO TEMPESTA

Disco a cura di
VITTORIO FERRAGUTI

Collaborano ad AmigaByte: Luca Arienti, Laura Baricevic, Paolo Bozzo, Luca Brigatti, Marco Brovelli, Paolo Colombo, Enrico Donna, Enrico Frascati, Renato Grossi, Fabrizio Lodi, Silvia Malaguti, Vincenzo Marangoni, Dario Martinelli, Luca Mirabelli, Lorenzo Orlandini, Roberto Pellagatti, Riccardo Premoli, Guido Quaroni, Fabio Rossetti, Emanuele Scribanti, Paolo Sisti, Ricky Sword, Mario Taddei, Aurora Tragara, Vertigo.

Redazione
C.so Vitt. Emanuele 15
20122 Milano
tel. 02/797830

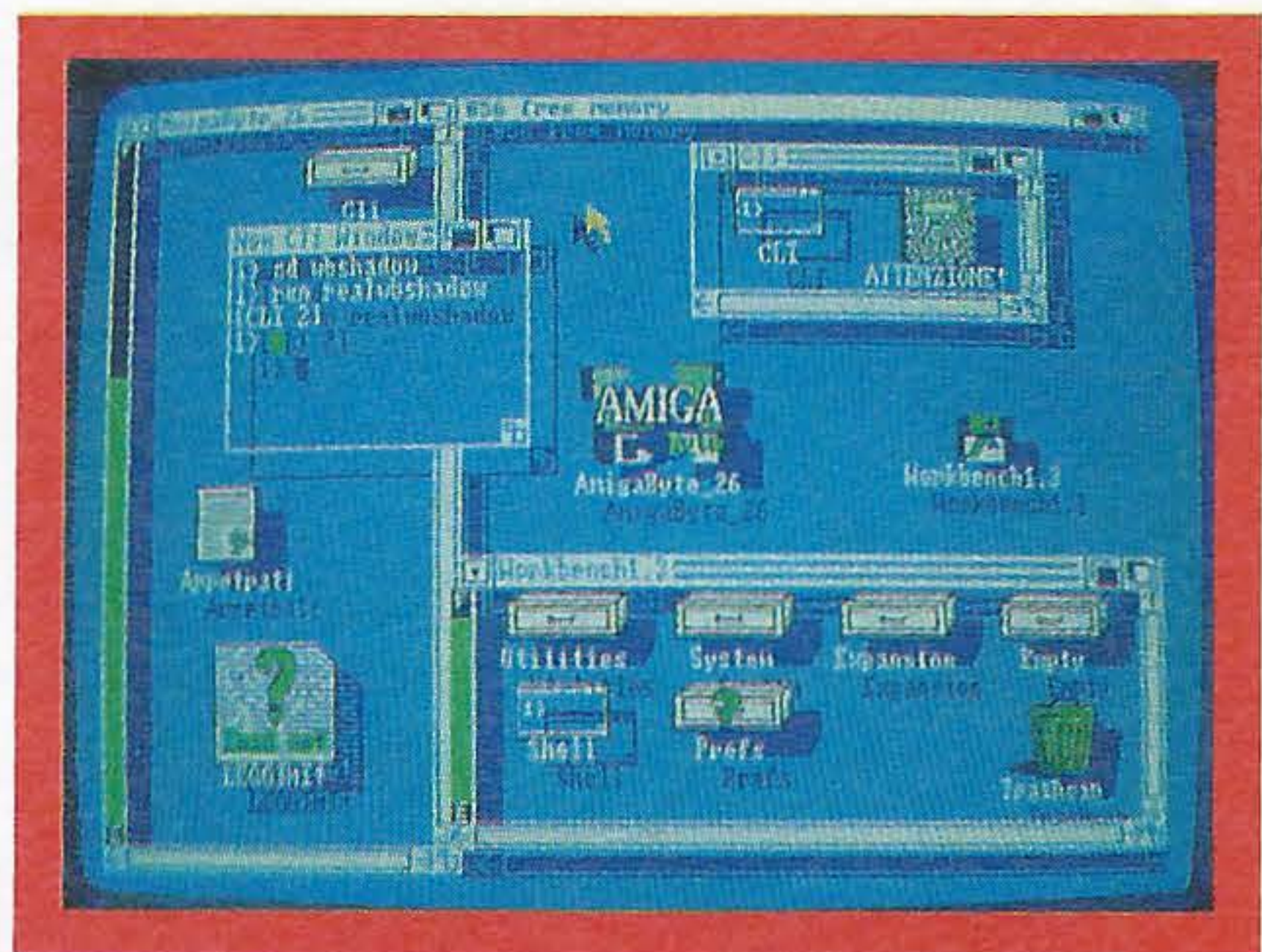
Amministrazione, Redazione, Pubblicità, Arcadia srl: C.so Vittorio Emanuele 15, 20122 Milano. Fotocomposizione: Compostudio Est, selezioni colore e fotolito: Eurofotolit. Stampa: Garzanti Editore S.p.A. Cernusco S/N (MI). Distribuzione: SO.DI.P. Angelo Patuzzi spa, Via Zuretti 25, Milano. Amiga Byte è un periodico mensile registrato presso il Tribunale di Milano al n. 215 il 29 marzo 1988. Resp. Sira Rocchi. Spedizione in abbonamento postale Gr. III/70. Pubblicità inferiore al 70%.

IL MENU

SUL DISCHETTO...

Il WorkBench, croce e delizia degli utilizzatori di Amiga, resta nonostante tutto il metodo più semplice ed immediato per sfruttare molte delle potenzialità del nostro computer preferito: a conferma di questa affermazione, ben tre dei programmi inclusi nel dischetto allegato al fascicolo di AmigaByte di questo mese sono dedicati a coloro che vogliono spremere al massimo le capacità di questa comoda interfaccia utente.
Il più spettacolare è certamente MEGAWB, un'utility pensata per coloro ai quali 640 pixel di larghezza per 256 di altezza sembrano troppo stretti. Grazie a questo programma i bordi dello schermo si dissolveranno come per magia, e l'ambiente WorkBench potrà estendersi fino alle dimensioni massime di 1024 per 1024 pixel, che potrete esplorare tranquillamente con il mouse. Mai più finestre sovrapposte e difficili da vedere: avrete spazio sufficiente per aprire tutte le finestre e gli schermi

che vorrete, memoria permettendo.
Per chi invece non ama pensare in grande o, più semplicemente, non vuole sprecare preziosa memoria chip, è stato ideato SMARTICON, che raggiunge lo stesso risultato attraverso un approccio diametralmente opposto: quando le finestre diventano troppo numerose, vi permette di farle scomparire temporaneamente (risparmiando perciò memoria) e di richiamarle all'istante solo quando serve, con un semplice click del mouse.

Siete incontentabili? Non vi basta aver aumentato larghezza ed altezza dello schermo con MegaWB? Tramite WBSHADOW potrete aggiungere una terza dimensione al WorkBench, ovviamente simulata ma non per questo meno gradevole: quella della profondità.
Dal WorkBench al CLI; il dischetto di questo mese comprende infatti due interessantissime utility che, per funzionare, richiedono l'uso di questo apparentemente ostile ambiente operativo. La prima, VSNAP, aggiunge al vostro mouse una straordinaria capacità: quella di tagliare ed incollare a piacimento porzioni dello schermo da un punto all'altro, eventualmente trasferendo i dati in esse contenuti tra programmi diversi. Se volete copiare parte di un'immagine, o una sezione di testo, dal display di un qualsiasi programma alla finestra del vostro word-processor preferito, non dovrete far altro che mettere mano al mouse e premere un tasto.
IFF2EXE è utilissimo invece a coloro che desiderano visualizzare immagini grafiche in formato Iff senza ricorrere alle tradizionali utility separate per il display: suo tramite è infatti possibile trasformare qualsiasi immagine in un singolo file eseguibile, e visualizzabile, autonomamente.
Ancora utility: TITANCRUNCH è un cruncher, uno di quei programmi che consentono di ridurre drasticamente l'occupazione di spazio di qualsiasi file, comprimendone il codice e riportandolo alla forma originaria prima di eseguirlo: la sua peculiarità consiste nel decomprimere i dati durate il caricamento da disco e non al suo termine, risparmiando tempo e soprattutto preziosa memoria.
Un tocco di allegria è rappresentato dal ritorno degli ANIM POINTERS, i puntatori animati già apparsi originariamente sul fascicolo 14 di AmigaByte: i nuovi soggetti di questa raccolta sono pesci, gatti ed astronavi in movimento.
L'ultima menzione, come di consueto, va al super gioco di questo mese: PONGO. Più che sulla grafica o sulla colonna sonora, l'accento è posto sul suo meccanismo di gioco, davvero coinvolgente. Un simpatico rompicapo popolato da labirinti e

barili da sistemare, che metterà a dura prova i vostri nervi ma dal quale non riuscirete a staccarvi facilmente: ve lo garantiamo.

Fino all'ultima nota!

«Soundtracker» ed «Oktalyzer» non rappresentano più l'ultima parola in fatto di musica. Ora ci sono «TFMX», «Future Composer» e «Delta Music», tre nuovi programmi che spremono il chip audio di Amiga fino all'ultima nota.

di ROBERTO PELLAGATTI

I primi programmi musicali per Amiga avevano in comune il metodo usato per l'inserimento delle note per la composizione dei propri brani musicali nella memoria del computer: **il pentagramma**.

Sebbene esso rappresenti il tipo di notazione migliore per musiche destinate ad essere lette ed eseguite con degli strumenti musicali, le cinque righe del pentagramma sono decisamente inadeguate qualora si desideri sfruttare appieno le possibilità di un computer.

Furono proprio queste limitazioni a contribuire all'affermazione di metodi alternativi di composizione musicale al computer: il primo programma ad introdurre un radicale mutamento di rotta nella creazione di musica con Amiga fu «**Soundtracker**», un software di pubblico dominio (di cui AmigaByte si è occupata dettagliatamente sul fascicolo di ottobre) che per primo ha abbandonato il pentagramma, come metodo per inserire le note, a favore dei **pattern**.

PERCHÉ IL PATTERN

La differenza, rispetto al pentagramma, è che nel pattern è possibile indicare, oltre che le note con le rispettive durate, anche un numero praticamente infinito di istruzioni tra le quali la scelta dell'inviluppo della nota stessa (ovvero il cambiamento nel tempo del suo volume), le modifiche nella sequenza, il canale audio da utilizzare, ed altre istruzioni ancora aventi, come ultimo limite, la fantasia del programmatore e la capacità di elaborazione del computer.

Anche i tre nuovi programmi che prenderemo in considerazioni utilizzano proprio il pattern come «strumento» per comporre. Questa scelta ovviamente penalizza chi è da sempre abituato ad usare la notazione classica e, di conseguenza, può scoraggiare quanti vorrebbero solamente giocherellare un po' con gli effetti sonori, ma è decisamente indicata per chi vuole fare seriamente musica con Amiga.

Basta comunque caricare uno dei dimostrativi forniti insieme ad ogni programma per riacquistare immediatamente tutto l'entusiasmo che solo Amiga sa regalare: suoni campionati e suoni sintetizzati, modulazione della voce per ottenere il «cantato», possibilità di includere i motivi musicali all'interno di propri programmi, sono solo alcuni fra i validissimi motivi che inducono a studiare a fondo il funzionamento di queste nuove creazioni.

ECCO A VOI «TMFX»

«**Tmfx**» è decisamente un programma innovativo: gli schemi introdotti da «**Oktalyzer**» e, in precedenza, da «**Soundtracker**», vengono qui potenziati grazie all'utilizzo, entro il pattern, delle **macro**. Con il termine «macro», in questo caso, si indicano generalmente quei comandi (o sequenze di comandi) defini-

bili dall'utente.

Prima di passare alle macro conviene però dare un'occhiata ai comandi utilizzabili entro il pattern. Ecco i codici esadecimali corrispondenti alle varie funzioni, affiancati da una breve spiegazione circa i loro effetti:

F0: END fine del pattern
F1: LOOP salto all'indietro
F2: CONT continuare
F3: WAIT attendere per un certo tempo
F4: STOP stop
F5: Kup ^ salire di tono
F6: VIBR vibrato
F7: ENVE controllo di inviluppo
F8: GSPT gosub (chiamata ad un altro pattern!!)
F9: ROPT return da chiamata
FA: FADE sfumare

Come si può vedere, sono presenti tutti i comandi

usuali con l'aggiunta dei **gosub** che, se opportunamente utilizzati, si rivelano preziosissimi per risparmiare tempo e strutturare meglio i pattern.

In alternativa ai comandi è possibile indicare un numero qualsiasi (sempre esadecimale) compreso tra 00 e 7F: ognuno di questi numeri indica una nota musicale.

Ogni riga del pattern, oltre che contenere il comando o la nota da eseguire, mette a disposizione degli spazi entro i quali bisogna inserire la macro da utilizzare (sulla quale torneremo molto presto), il volume per la nota, il canale ed il «detune» (modifica della frequenza base). Ogni comando necessita anche di alcuni parametri che vengono suggeriti direttamente dalla finestra di dialogo al momento del loro inserimento. Poiché i valori da dare a tali parametri risultano intuitivi, non è il caso di elencarli tutti.

Ecco ora una descrizione dettagliata del vero asso nella manica in dotazione a «Tfmx»: le macro.

LE MACRO DI TFMX

Visto che il semplice elenco dei comandi disponibili non basta a dare l'idea di come essi debbano essere utilizzati, si consiglia di studiare i demo forniti con il programma per vedere dei validi esempi.

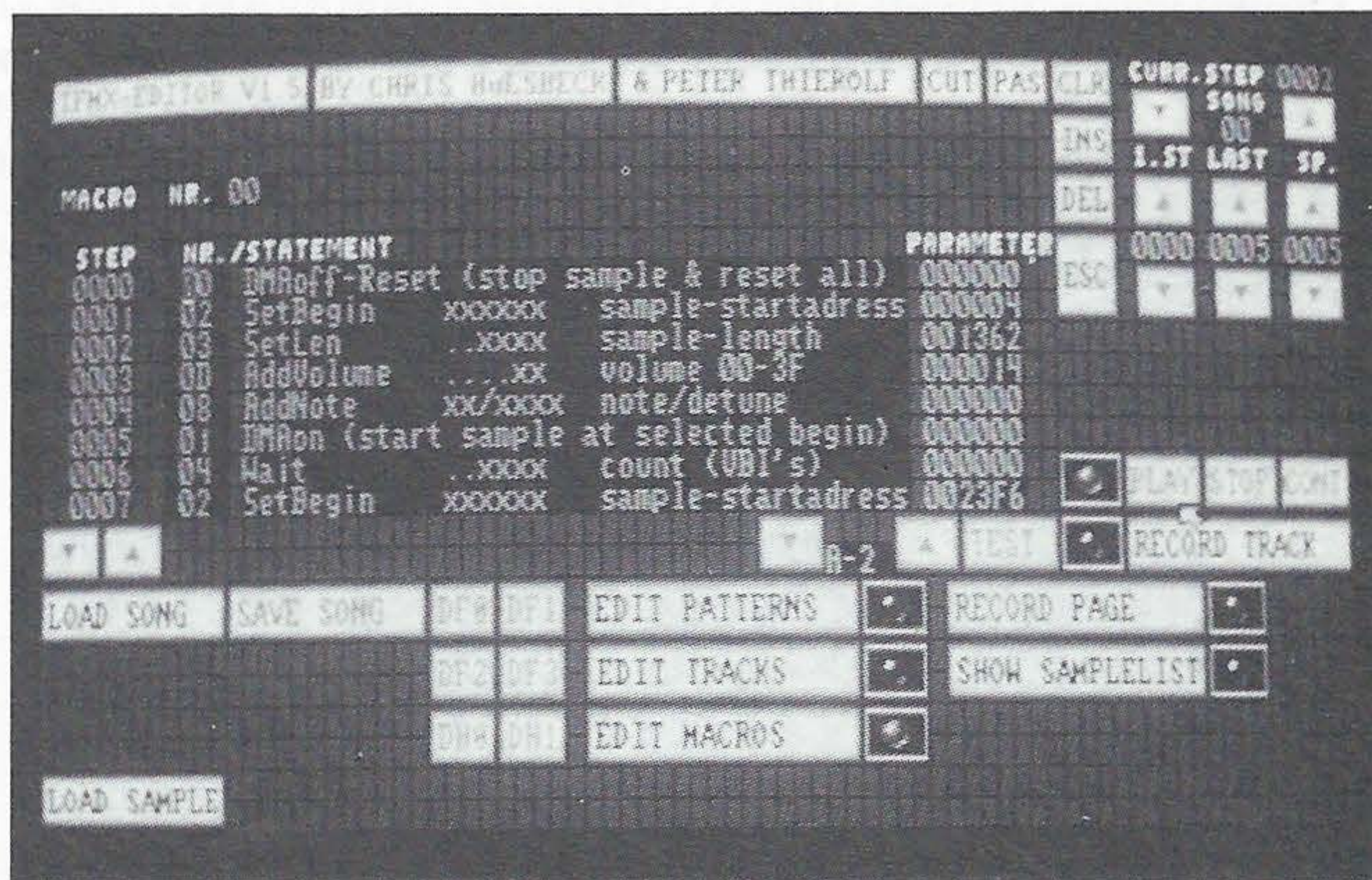
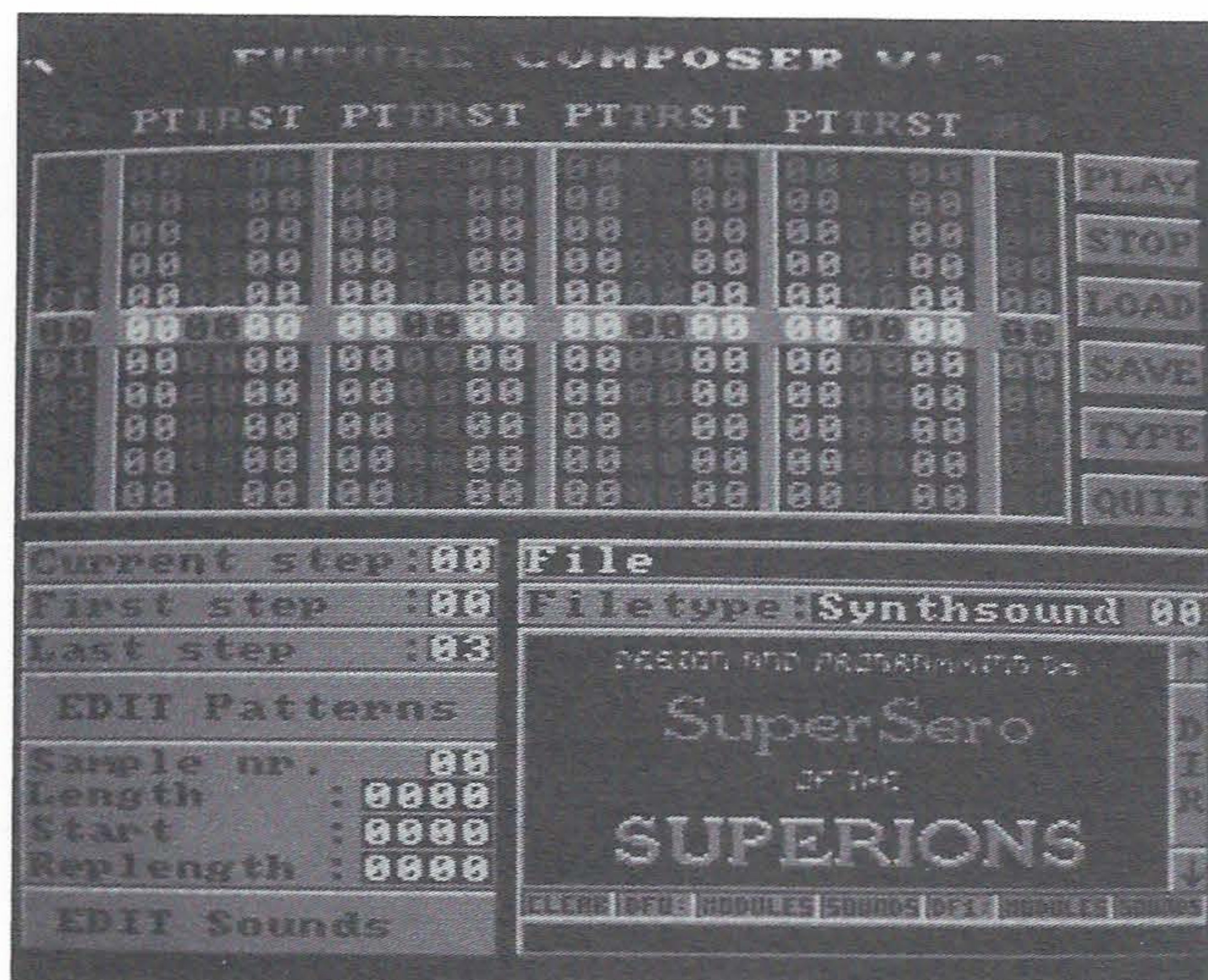
Entro la macro si può gestire in modo veramente completo uno strumento; è infatti possibile, anzi necessario, indicare l'indirizzo in memoria ove è contenuto il suono campionato, con i dati relativi ad uno strumento, la sua lunghezza, il punto in cui far iniziare il repeat; segue poi una serie di comandi per definire il portamento, il vibrato ed altri importanti parametri in modo molto più completo di quanto si possa fare con i soli pattern.

Giusto per dare un'idea delle possibilità offerte, si pensi alla modulazione della voce (una frase può essere utilizzata «cantando» note differenti) e degli strumenti (è possibile sentire l'imitazione di un sax veramente unica) e la creazione di suoni che cambiano forma d'onda mentre vengono suonati (con effetti simili a quelli forniti dai più potenti sintetizzatori!).

I comandi sono i seguenti:

00 = DMAOFF + RESET (Blocca il DMA e resetta i registri che lo controllano).
01 = DMAON (Attiva il DMA).
02 = SETBEGIN (Setta l'indirizzo di partenza del suono).
03 = SETLEN (Setta la lunghezza del suono).
04 = WAIT (Attende per un certo numero di cicli di clock).
05 = LOOP (Salta ad un determinato indirizzo !).
06 = CONT (Salta ad un indirizzo di un'altra macro !).
07 = ADDNOTE (Aggiunge uno o più semitoni).
08 = SETNOTE (Stabilisce la nota da suonare).
0A = RESET VIBRATO, PORTAMENTO, INVILUPPO
0B = SET PORTAMENTO
0C = SET VIBRATO
0D = ADD VOLUME
0E = SET VOLUME
0F = SET INVILUPPO
10 = LOOP KEY ON
11 = ADD BEGIN (Modifica l'indirizzo del suono campionato).
12 = ADD LEN (Modifica la lunghezza del suono campionato)
13 = DMAOFF NO CLEAR
14 = WAIT KEY UP
15 = GOSUB MACRO
16 = RETURN TO MACRO
17 = SET PERIOD
18 = SAMPLE LOOP
19 = SET ONESHOT SAMPLE

Riguardo all'uso di «Tmfx», sono poche le cose da rilevare. La presenza di numerosi gadget rende infatti superflue molte spiegazioni al punto che, a parte il significato dei vari comandi (comunque deducibili dai tanti demo), lo si potrebbe usare tranquilla-

Si notino le sequenze di numeri necessarie per definire i suoni in «Future composer». Sopra, ecco il generatore d'inviluppo; nella parte bassa dell'immagine, invece, il modulatore di frequenza.

Nell'immagine l'editor di «Tmfx» per l'assegnazione delle macro di un brano musicale. La seconda colonna contiene il numero del comando.

«Future composer» è un valido concorrente di «Tmfx». La presentazione grafica è ben curata. È possibile, tramite gli appositi gadget, selezionare altri screen per l'editing dei brani.

mente anche senza l'aiuto del manuale.

Unica avvertenza: prima di suonare un motivo bisogna ricordarsi di togliere il metronomo, altrimenti un noiosissimo ticchettio rovinerebbe l'esecuzione.

Un'ultima peculiarità veramente inedita caratterizza «Tfmx», e cioè la possibilità di inserire le note nei pattern suonandole in tempo reale con i tasti della tastiera (come accade con «**Sonix**»), con il ritmo scandito dal metronomo.

Peccato che questa caratteristica, che da sola già renderebbe unico il programma, passi quasi inosservata se paragonata alle altrettanto notevoli prestazioni prettamente sonore!

L'aver parlato subito di «Tmfx» non deve far pensare che «Delta-Music» (d'ora in poi «DM») e «Future Composer» (d'ora in poi «FC») siano da meno.

Questi due programmi infatti, pur appartenendo alla categoria del software di pubblico dominio/shareware, hanno caratteristiche tali da reggere bene il confronto con il «fratello maggiore» di origine commerciale.

FUTURE COMPOSER

Tra i due, il primo apparso è «**Future Composer**».

«FC» possiede tutte le caratteristiche fondamentali per proporsi come alternativa ad altri programmi già in commercio o di pubblico dominio. L'inserimento dei dati relativi ai brani da comporre avviene attraverso vari schermi dedicati ai pattern, ai suoni, ed alla gestione globale di tutte le opzioni generali.

Pur essendo leggermente superiore a «Soundtracker» in tutte le opzioni fondamentali, le sue caratteristiche, se si fermassero qui, non basterebbero da sole a giustificare il passaggio ad «FC» da parte di chi era già abituato a lavorare nell'altro ambiente. È quindi doveroso citare il vero punto di forza che lo trasforma in un pericoloso concorrente: **i moduli** (ovvero i file contenenti i brani musicali composti) che esso genera risultano molto più corti, con evidente risparmio di memoria quando devono essere inglobati in demo o, comunque, all'interno di altri programmi.

«FC» infatti, al pari di «Soundtracker», viene fornito insieme ad un listato assembler incorporabile in qualsiasi altro programma scritto dall'utente in quel linguaggio, che permette di suonare i sopracitati moduli. Questo consente al programmatore di concentrarsi sugli aspetti «artistici» della parte musicale senza dover perdere tempo a scrivere il codice relativo alla sua esecuzione.

«FC» riesce a risparmiare spazio sia nella gestione dei pattern che in quella dei suoni. I primi, grazie ad istruzioni molto brevi e potenti, permettono di descrivere in modo sintetico la melodia, mentre i secondi si avvalgono di una caratteristica nuova (escludendo «Sonix»): la possibilità di sintetizzare dei suoni mentre vengono suonati, senza ricorrere ai sample (forme d'onda campionate) la cui occupazione in termini di spazio è, notoriamente, sempre misurabile in decine di «kappa».

In questo modo si utilizzano i suoni campionati solo per descrivere gli strumenti percussivi (notoriamente molto brevi) mentre per strumenti quali l'organo, il piano, gli ottoni e tutti i suoni «digitali», si può tranquillamente ricorrere al modulo sintetizzatore nel quale si descrive la forma d'onda da creare per via parametrica e non tramite il campionamento.

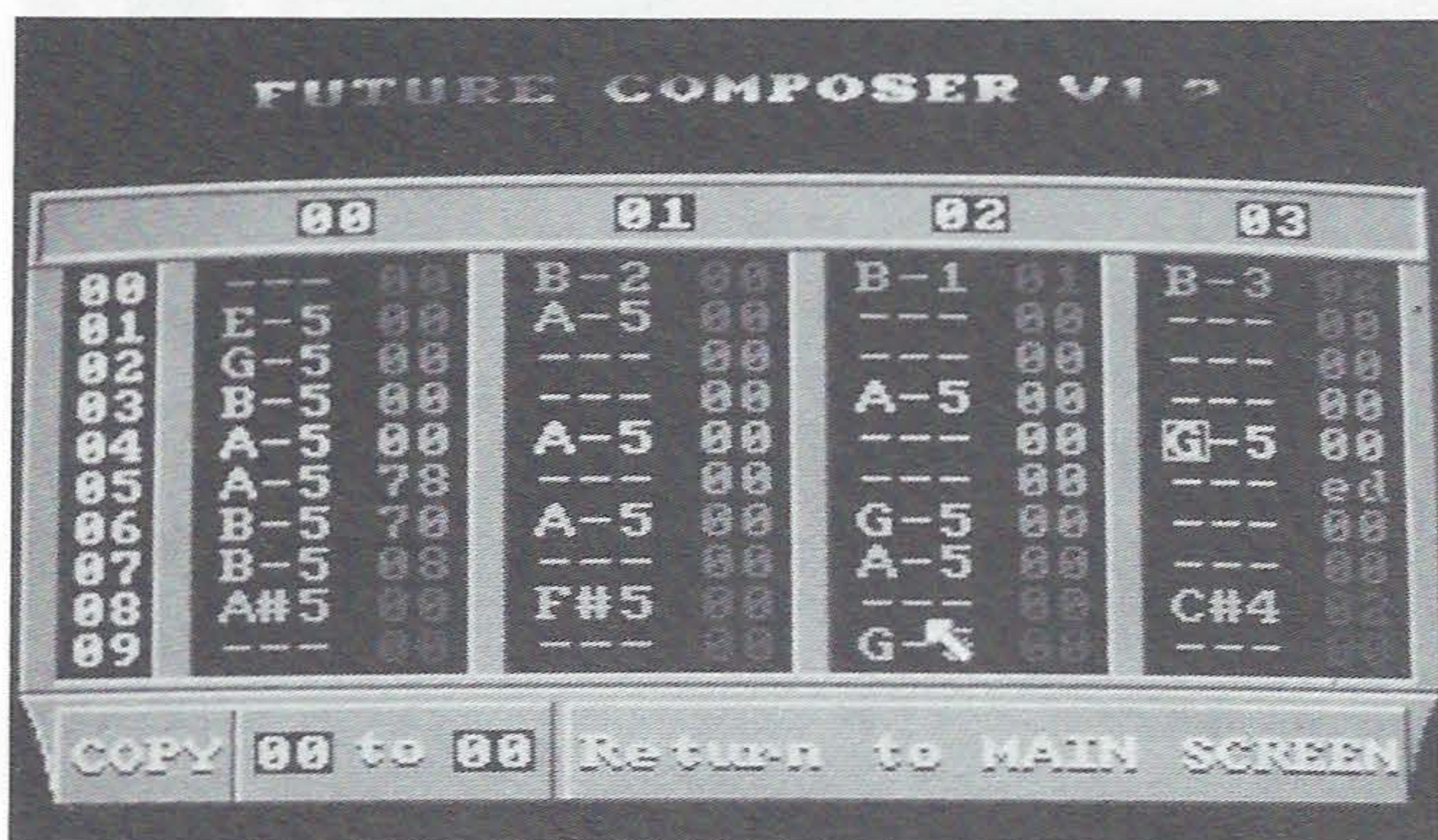
Utilizzare «FC» è abbastanza semplice anche se, per alcune funzioni, bisogna ricorrere alla tastiera; ogni pattern scritto con «FC» fa riferimento ad un unico canale audio, a differenza di «Soundtracker» dove un pattern interessava tutti e quattro i canali.

Grazie a questo accorgimento è possibile evitare inutili ripetizioni di dati, soprattutto per quanto riguarda le traccie ritmiche, nelle quali si ripete il medesimo schema di fondo (il tempo dato dalla batteria), mentre cambia solo la parte melodica. Come di consueto, ogni posizione entro il pattern può contenere il codice della nota da suonare oppure un comando.

Per incrementare o decrementare una nota (ad esempio da Do a Do# o Si) si possono usare i tasti di funzione F6 e F7, mentre F9 e F10 servono ad inserire o ad eliminare una nota. Per copiare un pattern è sufficiente inserire i pattern sorgente e destinazione a destra dell'icona «Copy», e poi clickare sull'icona stessa.

LE NOTE MUSICALI

Le note musicali sono attivate tramite i tasti «qwertyu» e «23567» (con ovvio significato per quanto riguarda la loro posizione). I tasti da F1 a F4 servono per scegliere l'ottava, mentre le coppie di tasti 7 e 8, 4 e 5, 1 e 2, 0 e «.» del tastierino numerico servono per decrementare od incrementare il numero del pattern corrispondente (da uno a quattro).

Ancora più interessante

Anche l'inserimento dei pattern necessita, in «Future composer», di uno schermo apposito. Dopo un po' di pratica il suo utilizzo risulta abbastanza veloce.

è l'editor dei suoni, poiché presenta caratteristiche davvero nuove: nello screen relativo risaltano subito due finestre con la scritta «**Volume Modulation Sequence**» e «**Frequence Modulation Sequence**».

Come rivelano gli stessi nomi, in queste finestre vanno inseriti i parametri necessari per controllare l'inviluppo e la modulazione di frequenza dei suoni; entrambe sono caratterizzate da una serie di numeri rappresentanti ognuno un byte in notazione esadecimale.

Per quanto riguarda la Volume Modulation Sequence, questo è il significato dei numeri impiegati:

Byte 0 : velocità di esecuzione della sequenza;
Byte 1 : numero della Frequence Modulation Sequence da usare;
Byte 2 : velocità del vibrato;
Byte 3 : ampiezza del vibrato;
Byte 4 : delay prima dell'esecuzione del vibrato;
Byte 5 : volume iniziale.

I byte seguenti sono i dati: i valori compresi tra le posizioni 6 e 63 indicano il volume. Sono disponibili anche i seguenti comandi:

END : byte di valore E1;
LOOP : byte di valore E0 (il byte successivo indica la posizione cui saltare);
SUSTAIN : byte di valore E8 (il byte successivo indica per quanto tempo deve durare il sustain).

Questi sono invece i corrispondenti dei byte nella Frequence Modulation Sequence. Ogni sequenza deve iniziare con i valori «E2 xx» o «E4 xx», dove «xx» è il numero di una forma d'onda. I numeri da $00 a $09 sono riservati ai sample, mentre quelli da $0a a $36 sono per i suoni sintetizzati secondo il seguente schema:

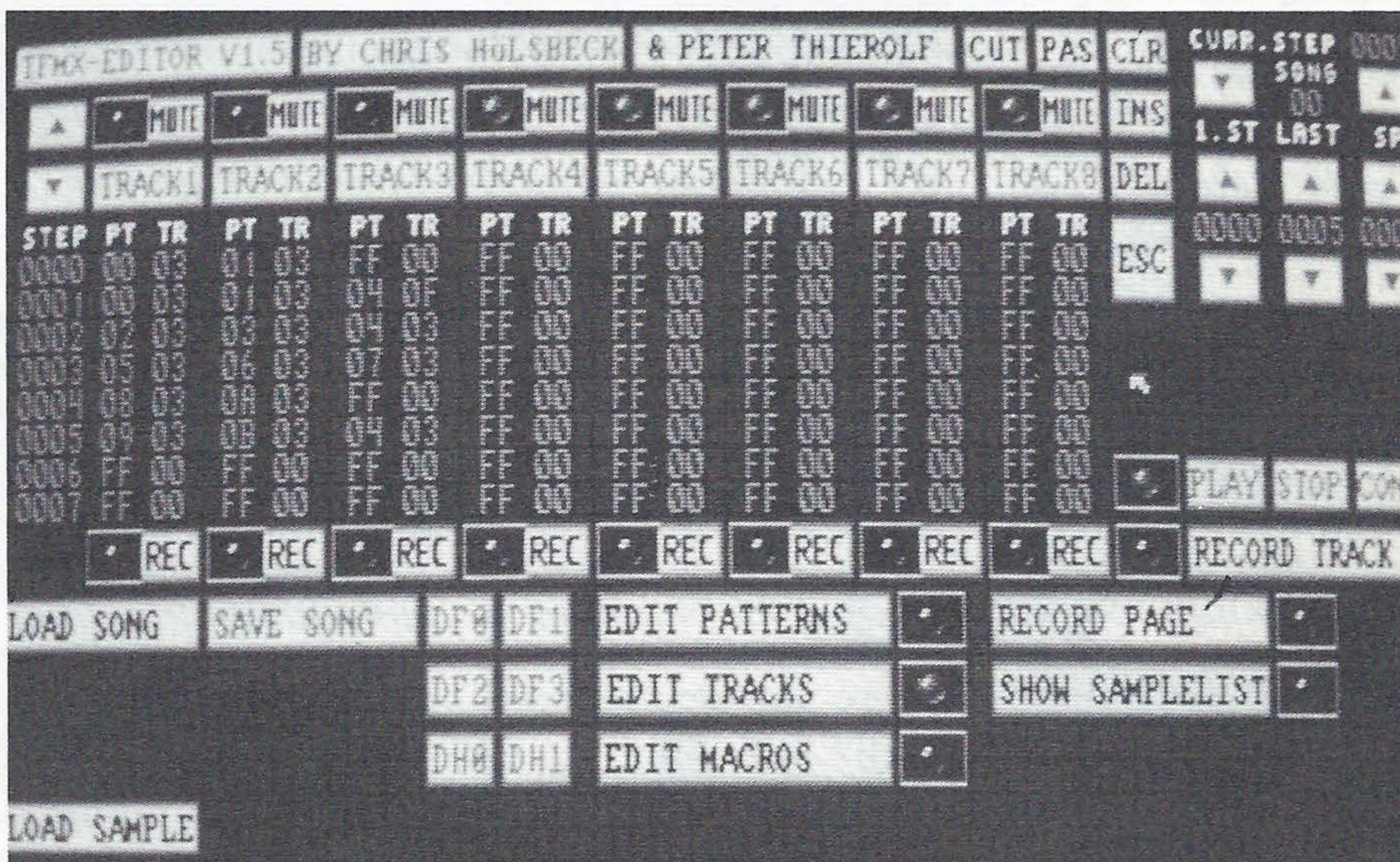
$0a-$19 = Onde sinusoidali.
$1a-$31 = Onde quadre.
$32-$33 = Onde a dente di sega.
$34-$36 = Non definite.

Tutti i byte dalla posizione 2 alla 63 sono dei byte di Transpose (shift della frequenza), tranne i seguenti comandi:

END Byte di valore E1.
LOOP Byte di valore E0.
SET WAVEFORM Byte di valore E2: il byte successivo indica la forma d'onda.
CHANGE WAVEFORM Byte di valore E4: il byte successivo indica la nuova forma d'onda.
NEW VIBRATO Byte di valore E3: i due byte successivi indicano la velocità e l'ampiezza.
SUSTAIN Byte di valore E8: segue il tempo di sustain.
NEW SEQUENCE Byte di valore E7: permette di passare ad un altro numero di sequenza, funzionando in pratica come un comando «goto».

La descrizione di questi codici può far sembrare la composizione delle musiche con «FC» più complessa di quanto non sia realmente; per rendersi conto

PATTERN O PENTAGRAMMA?

Chi è abituato ad usare il pentagramma per scrivere la musica, è pronto a giurare che si tratta del metodo ideale per la composizione, visto che rappresenta il risultato di secoli di studi e di modifiche.
In realtà le obiezioni da fare sono molte: anzitutto bisogna dire che la notazione musicale tradizionale spesso è inefficace anche quando viene usata nelle partiture per alcuni strumenti «classici». Un buon saxofonista, ad esempio, per poter suonare un brano scritto su pentagramma deve prima di tutto interpretare il brano per capire come deve essere suonato. Ogni singola nota, infatti, potrebbe essere resa con una infinità di sfumature dipendenti dall'«atmosfera» del brano stesso. Anche le partiture per batteria sono alquanto difficili da

Ecco come si presenta in «Tmfx» la sequenza di istruzioni necessaria per suonare uno dei tanti motivi dimostrativi: una bella differenza rispetto al pentagramma!

scrivere, e pochi batteristi sanno leggere uno spartito per il loro strumento.
Ovviamente, grazie anche all'uso di alcune notazioni supplementari, il pentagramma resta il mezzo più valido e generale per scrivere brani destinati ad essere suonati da un essere umano.
Quando però l'esecutore è il computer, le cose cambiano: poiché non si può pretendere che la macchina «capisca» il pezzo in modo da darne una propria interpretazione, è indispensabile ricorrere a qualche tipo di notazione più efficace per sfruttarne le capacità. Purtroppo le caratteristiche dei computer, considerando tali anche i vari sequencer, sintetizzatori e così via, sono tanto differenti che è difficile creare una notazione standard valida per tutti (è altresì difficile creare uno standard nell'ambito del software per uno stesso modello di computer!).
Per ora è il pattern, in tutte le sue forme e variazioni, ad essere generalmente utilizzato, in quanto rappresenta il miglior compromesso fra leggibilità (qualcuno deve pur scrivere i brani per il computer!) e potenza d'uso per quanto riguarda lo sfruttamento delle capacità degli elaboratori ed il superamento dei loro limiti.

Guardando lo screen di «Delta Music» si nota subito che l'interfaccia grafica potrebbe essere migliorata. Risalta comunque immediatamente anche l'editor delle forme d'onda.

in pratica con alcuni esempi, è sufficiente caricare i brani forniti con il programma.

Come già accennato, insieme a «FC» viene fornita una **routine-player** dei brani musicali, pronta per essere assemblata (con il Seka assembler) o inclusa nei propri codici sorgente. I programmatori sappiano che per utilizzarla è sufficiente, con un programma esterno, ricorrere alle chiamate:

Jsr INIT_MUSIC: per l'inizializzazione delle variabili;

Jsr PLAY_MUSIC: da usare una volta ogni cinquantesimo di secondo (è sufficiente sincronizzarsi con il display);

Jsr END_MUSIC: per finire e disallocare la memoria utilizzata dalla routine.

È LA VOLTA DI DELTA-MUSIC

«**Delta-Music**» (d'ora in poi «DM»), scritto da Bent Nielsen, nasce come possibile sostituto di «FC», come dichiarato da Nielsen stesso, in quanto con esso è possibile definire **le forme d'onda** tracciandole **con il mouse** anziché utilizzando serie di numeri inseriti manualmente.

Sotto questo punto di vista, «DM» è davvero superiore a «FC»; purtroppo l'interfaccia-utente del programma è alquanto scadente, forse perché non è stata scritta in previsione di una distribuzione commerciale su larga scala del software. Ciò nonostante, considerate le sue caratteristiche positive, si può anche essere disposti a «soffrire» un po'. Il «**track editor**» è basato su di una sequenza di coppie di byte: il primo byte della coppia indica il blocco che si desidera suonare (in «DM» il blocco rappresenta l'equivalente del pattern), mentre il secondo indica il numero di semitoni da sommare o sottrarre a tutte le note del blocco indicato (**key transpose**).

Per indicare la fine di un brano basta assegnare il valore $FF ad entrambi i byte nella posizione voluta; i due byte successivi dovranno indicare la linea cui saltare per ricominciare il brano.

Il «**block editor**» permette invece di inserire le note o i comandi. Ogni linea di un blocco è costituita da quattro valori che indicano rispettivamente la nota da suonare, lo strumento da utilizzare, il codice dell'effetto, ed un parametro relativo all'effetto.

$00 = normal play (no effect)
$01 = set new playspeed
$02 = slide frequency up
$03 = slide frequency down
$04 = set LED (power light)
$05 = set new vibrator wait
$06 = set new vibrator step
$07 = set new vibrator length
$08 = set new bendrate
$09 = set new portamento
$0A = set new volume
$0B = set new arpeggio byte 1
$0C = set new arpeggio byte 2
$0D = set new arpeggio byte 3
$0E = set new arpeggio byte 4
$0F = set new arpeggio byte 5
$10 = set new arpeggio byte 6
$11 = set new arpeggio byte 7
$12 = set new arpeggio byte 8
$13 = set new arpeggio byte 1 & 5
$14 = set new arpeggio byte 2 & 6
$15 = set new arpeggio byte 3 & 7
$16 = set new arpeggio byte 4 & 8
$17 = set new attack step
$18 = set new attack delay
$19 = set new decay step
$1A = set new decay delay
$1B = set new sustain high byte
$1C = set new sustain low byte
$1D = set new release step
$1E = set new release delay

I codici esadecimali per l'inserimento di effetti sonori in un blocco di «Delta-Music»

Poiché «DM» non possiede le macro come «Tfmx», l'autore ha evidentemente cercato di compensarne la mancanza mettendo a disposizione una notevole quantità di effetti predefiniti. L'elenco completo di questi effetti e dei codici necessari ad ottenerli è riportato in figura 1.

L'editor dei suoni è altrettanto ricco di comandi, il cui significato è sufficientemente intuitivo da consentirci di tralasciare l'elenco. L'unica particolarità degna di nota risiede nel fatto che se, al posto del nome dello strumento, si scrive «—», il suono non verrà caricato da disco.

Il generatore di suoni è composto dalla lista di istruzioni necessarie per modularli; le istruzioni sono codificate per mezzo di una serie di byte.

I valori compresi fra $00 e $0F servono ad indicare il numero della forma d'onda (ovvero lo strumento) da utilizzare; i valori fra $80 e $FE, invece, indicano la velocità con cui bisogna eseguire i comandi nella lista: $80 indica la massima velocità e $FE la minima. Con $FF si indica la fine della lista, ed il byte successivo serve per indicare la riga alla quale bisogna saltare per eseguire il repeat delle istruzioni.

QUALE SCEGLIERE?

Tutti i programmi qui presentati possiedono caratteristiche superiori a quelle della maggior parte dei programmi musicali per Amiga finora esistenti; l'unica eccezione è costituita da «Oktalyzer», già recensito sul numero 20 di AmigaByte, il quale, grazie alla possibilità di usare ben 8 canali audio e di campionare direttamente i suoni, rimane sempre un validis-

simo concorrente.

«TMFX» è sicuramente un punto di riferimento per tutto il software esistente e per quello che verrà; trattandosi di un programma commerciale, i suoi autori dovevano andare a colpo sicuro per invogliare gli utenti Amiga ad acquistarlo.

Bisogna ammettere che ci sono davvero riusciti: l'interfaccia grafica molto potente ed user-friendly, la possibilità di scrivere i pattern suonando le note in tempo reale e la presenza delle macro lo rendono decisamente attraente. Non ci sarebbe poi da stupirsi se qualcuno lo comprasse anche solo per poter sentire i fantastici demo di cui è dotato! L'unico neo è la facilità con cui va in Guru quando si inseriscono sequenze di comandi illogiche, o si salta troppo frequentemente da uno screen all'altro.

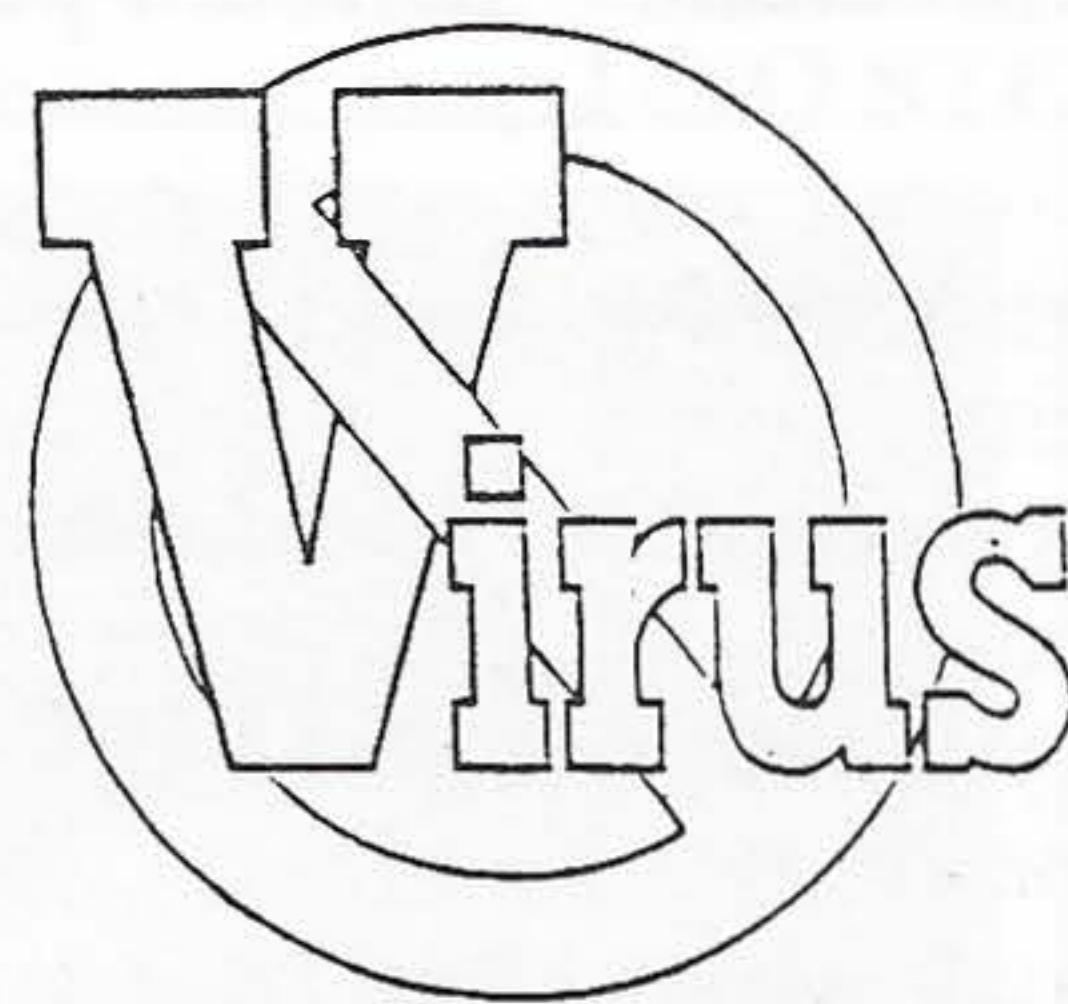
Non è ovviamente il caso di chi utilizza il programma seriamente, ma d'altra parte un programma commerciale dovrebbe essere a prova d'errore anche quando si inseriscono comandi «a caso».

«Future Composer» e «Delta-Music», gli altri due programmi appartenenti al mondo del software PD, godranno probabilmente per questo motivo di una maggiore diffusione ed essendo dotati, al pari del «Soundtracker», di un player scritto in assembler pronto per essere usato in altri sorgenti, verranno sicuramente utilizzati dai programmatori più esperti.

«Delta-Music» sembra essere il più potente fra i due; peccato che l'interfaccia-utente sia piuttosto mediocre, al punto che l'autore si dichiara, nella documentazione, disposto a cedere il sorgente a chi abbia tempo e voglia di preparare una nuova release migliorata sotto questo profilo.

Chi non ha una certa predisposizione per la musica troverà serie difficoltà nel capire come usare questi programmi ed anche coloro che conoscono benissimo il pentagramma, abituati a scrivere musica o magari diplomati al Conservatorio, si ritroveranno un po' spaesati e preferiranno utilizzare il «Deluxe Music».

In effetti, fare musica con questi programmi, anziché utilizzare i più tradizionali «DMCS» o «Sonix», è un po' come programmare in Assembler piuttosto che in Basic: ciò che si guadagna sotto il profilo del controllo totale dell'hardware del computer, si perde in immediatezza e semplicità.

Coloro che, invece, sono già abituati ad usare pattern, suoni campionati, sintetizzatori o routine sonore «self-made», si troveranno a proprio agio, ovviamente dopo aver dedicato un po' di tempo alla lettura dei manuali d'uso ed all'analisi dei numerosi esempi, davvero preziosi.

□

Effetti ed animazioni con TurboSilver

Si conclude il nostro corso di ray-tracing con «TurboSilver», analizzando le funzioni di animazione e gli effetti speciali ottenibili tramite l'uso di brush e stencil.

di ENRICO FRASCATI
Terza parte

In questa puntata conclusiva del nostro lungo e completissimo tutorial sull'uso del potente software grafico «**TurboSilver**» ci occupiamo di tutte le funzioni e dei comandi tralasciati in precedenza: l'editor degli oggetti, la funzione di estrusione, l'applicazione di brush e stencil, e la creazione di animazioni in ray-tracing.

Ricordiamo che le altre funzioni di «TurboSilver», necessarie per la comprensione di questo articolo, sono state spiegate in dettaglio nei fascicoli 23 e 24 di AmigaByte.

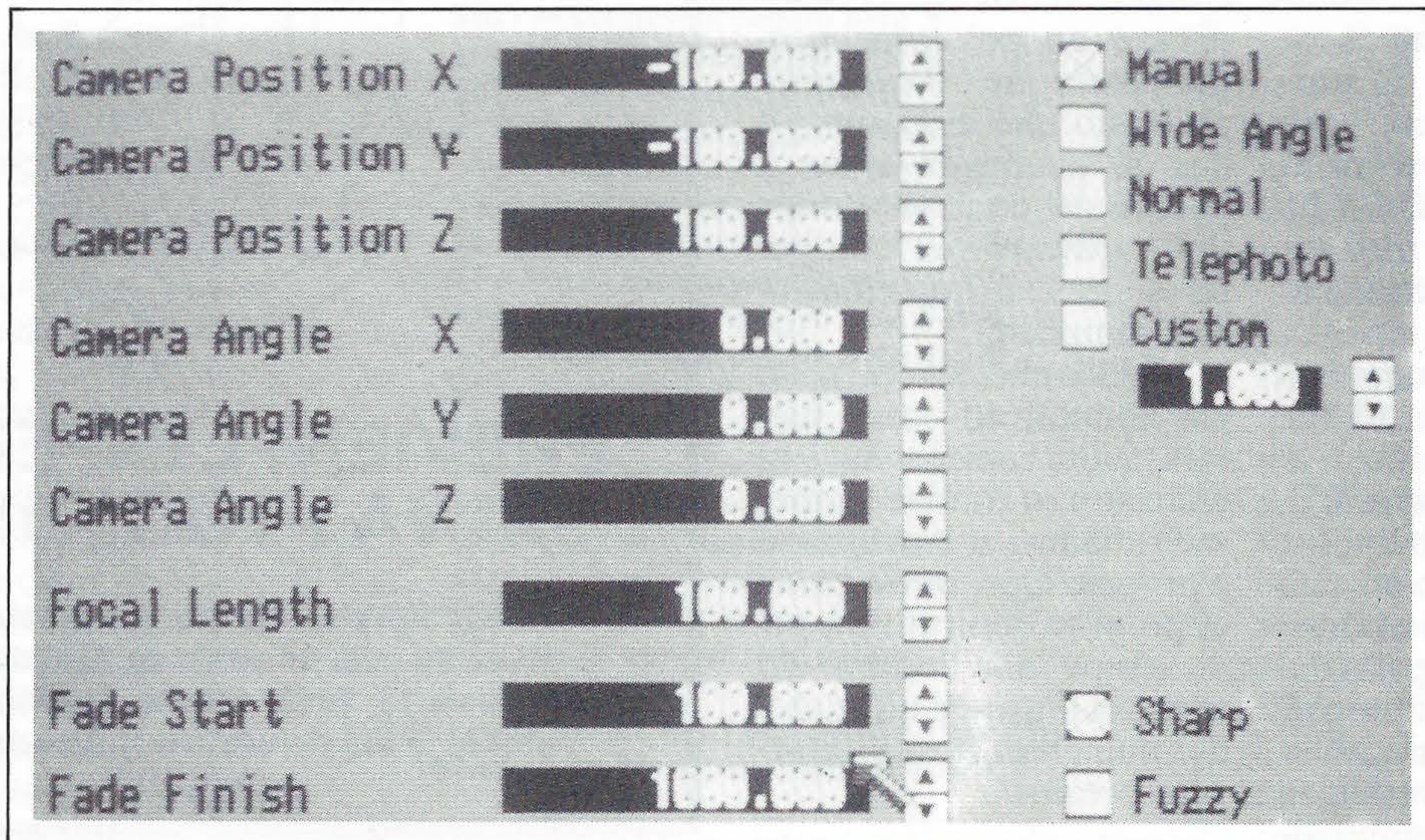
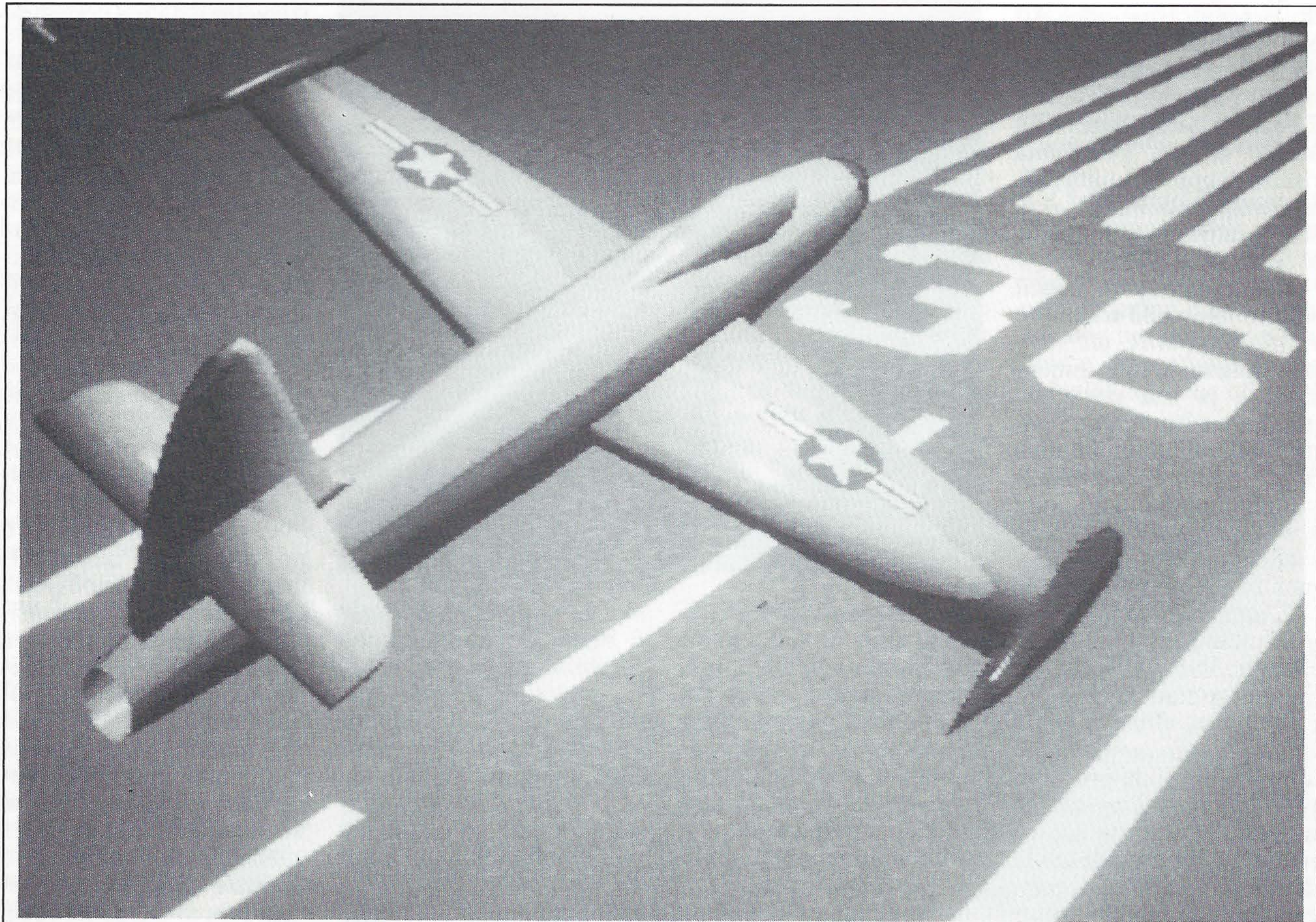
Abbiamo già avuto occasione di rilevare come l'editor degli oggetti di «Turbo-Silver» (d'ora in poi «TS») non sia tra i più versatili, consigliando a chi ne ha la possibilità l'uso di programmi dedicati come «**Modeler 3d**»; può tornare comunque utile saper usare le funzioni di editing interne di «TS», quindi dedichiamo un po' di spazio all'argomento.

Per creare un oggetto da zero è necessario avere un asse al quale agganciare i suoi punti. Entrate quindi nell'editor di cella e selezionate l'opzione «**Add Axis**» dal **menu «Edit»**. Apparirà quindi un asse «nudo»: selezionatelo, ed usate «**Add Point**» per aggiungere alcuni punti attorno all'asse. Attivate poi «**Add Face**» e, tre alla volta, connettete i punti per formare la faccia dell'oggetto.

Ricordate che per vedere un oggetto dopo il rendering è necessario che esso sia formato da facce: nessun altro elemento viene visualizzato.

Inizialmente si procede creando un semplice oggetto piano, e lavorando sui diversi piani cartesiani, potrete elaborarlo fino a trasformarlo completamente. «TS» fornisce dei tool che permettono di svolgere queste operazioni in modo automatico: selezionando l'oggetto scegliete «**Facemenu** «**Pick**». In questo modo l'editor controllerà solo le facce che avete creato fra i punti.

Dal **menu «Edit»** selezionate poi «**All**» e «**Delete**» ed avrete così «ripulito» i punti dell'oggetto; con «**Add Edge**» connetteteli poi nuovamente per formare un contorno chiuso.

Dal **menu «Settings»** scegliete poi «**Mold**» dopo esservi assicurati che l'oggetto sia selezionato. La finestra che appare è usata per tutti i tipi di elaborazioni geometriche degli oggetti permesse da «TS»: estrusione (**Extrude**) in lunghezza (**To Lenght**), estrusione lungo un percorso (**By Story**), rotazioni attorno all'asse (**Spin** e **Sweep**). È possibile inoltre decidere se l'oggetto risultante debba avere gli estremi aperti (**Open**) o chiusi (**Closed**); stabilire la lunghezza

dell'oggetto estruso (**Lenght**), l'angolo in gradi della rotazione (**Angle**), ed il numero di sezioni da cui deve essere composto (più sono, più l'oggetto è definito e lungo da elaborare; è possibile ovviare al problema utilizzando poche sezioni e l'opzione «**Smooth**» dalla finestra di «**Attributes**»).

Provate subito un'estrusione in lunghezza del vostro oggetto; per annullare il risultato dell'operazione potete selezionare «**Undo**» dal **menu «Edit»**.

GLI OGGETTI SIMMETRICI

«Sweep» e «Spin» hanno un'importanza fondamentale nella costruzione di oggetti simmetrici come vasi, bottiglie o colonne: create un asse e, osservandolo con la «**Front View**», immaginatelo come il centro dell'oggetto. Posizionate quindi, lateralmente all'asse, alcuni punti per indicare il contorno; collegateli con «**Edge**» e fate uno «**Sweep** per procedere al tracciamento.

Per quel che riguarda lo «**Spin**», invece, la cosa è un po' diversa: verranno infatti ruotati tutti i punti eccetto che il primo e l'ultimo, creado forme astratte, vagamente a goccia.

L'unico comando della finestra non ancora spiegato è quello relativo all'estrusione lungo un percorso. Esso è collegato alla voce seguente del menu: «**Story**».

Il comando «**Story**» è importante non solo per le estrusioni ma anche, e principalmente, per le animazioni. Permette infatti di assegnare ad un oggetto un percorso (Path) o un movimento primitivo (rotazione e cambiamento in scala delle dimensioni, per simulare certi effetti di prospettiva).

Per creare un path è sufficiente posizionare una serie di punti e connetterli con «**Edge**». È necessario però che ogni path abbia un nome proprio: per questo dovrete cambiare quello di default («**axis**») con uno a vostra scelta, tramite il **menu «Attributes»**.

Selezionate quindi l'oggetto da estrudere e l'opzione «**Story**»: alla richiesta «**Follow Path**» inserite il nome che avete dato al path, ed il gioco è fatto. A questo punto, utilizzando la funzione «**Extrude by Story**», l'oggetto si allungherà seguendo il percorso che gli avete preparato.

Nella finestra «Story», in basso a sinistra, c'è un altro selettore importante, «**Y Align**», selezionando il quale farete si che l'oggetto ruoti il suo asse seguendo quello del path. Se la cosa non vi sembra chiara, fate una prova e capirete subito di che si tratta; ricordate però che per usare questo comando il path deve essere steso sul **piano Y**. La funzione «Story» sarà ripresa più avanti parlando delle animazioni.

CREARE UNO STENCIL

Gli «**stencil**» sono immagini pittoriche in due colori, uno di sfondo (**background**) ed uno del pennello (**foreground**), che «TS» importa come veri e propri oggetti. Una volta eseguito il rendering, sarà visibile solo il colore di foreground dello stencil, mentre il background risulterà essere totalmente trasparente.

Uno stencil non può essere estruso, ma il foreground può essere modificato come la superficie di un normale oggetto, e reso speculare, luminoso, ruvi-

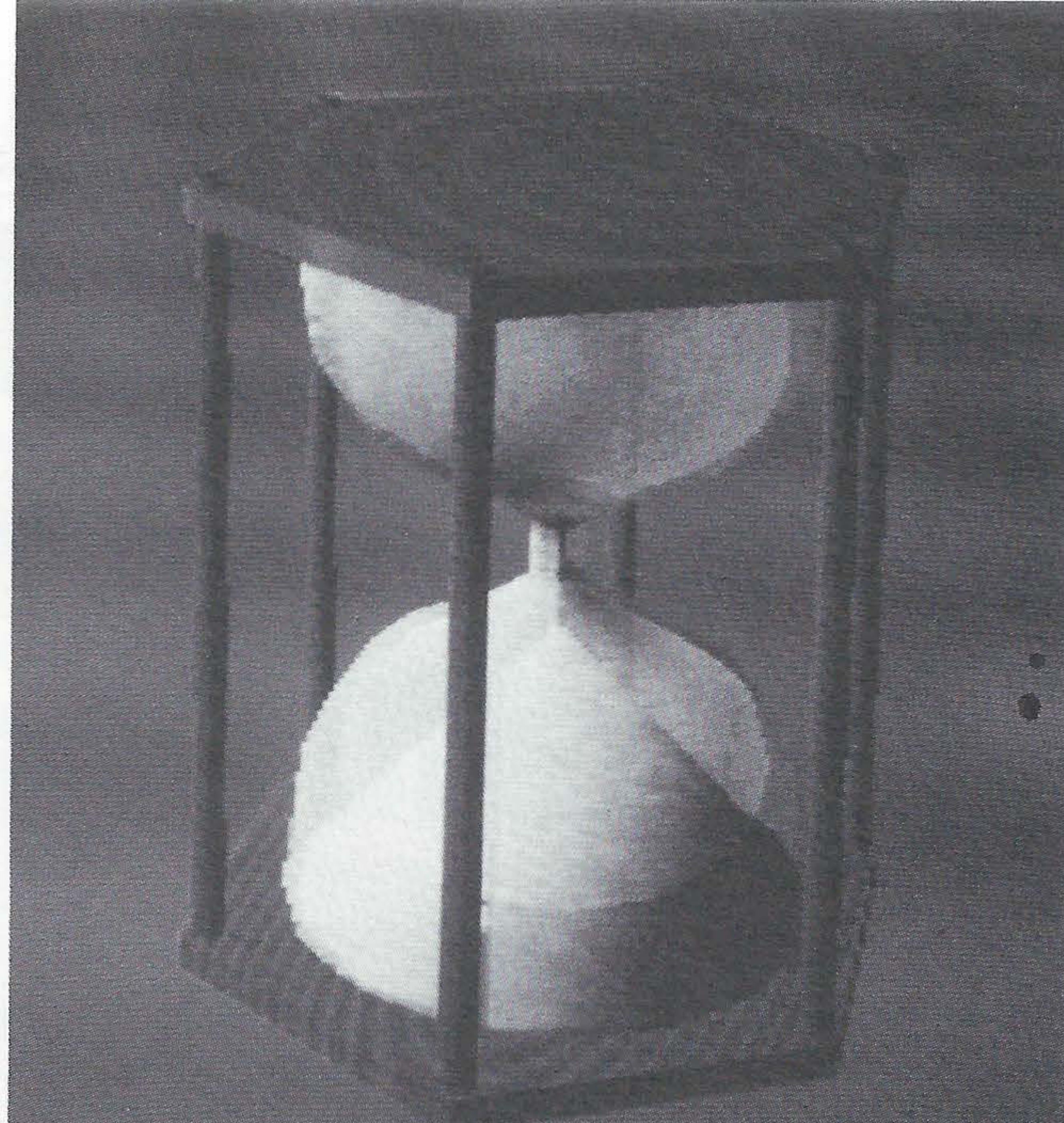
do o semitrasparente. Ad esso può anche essere applicata una **texture** (ne abbiamo parlato sui fascicoli 23 e 24 di AmigaByte) o un **brush**. Questa possibilità risulta di grande utilità nella creazione di marchi o di scritte, altrimenti di difficile editazione.

Per creare uno stencil selezionate «**Add stencil**» e vedrete apparire un semplice piano; le sue dimensioni e la sua posizione possono essere variate come quelle di qualsiasi altro oggetto. Potrete applicarvi l'immagine da voi scelta andando al **menu «Stencil»** e dando il comando «**Load**».

L'immagine può essere vista dando il comando «**View**». Può capitare che questo comando distorca l'immagine, visualizzandola nella risoluzione sbagliata, ma non c'è da preoccuparsi: lo stencil verrà comunque corretto in fase di rendering. Ricordate che la directory dalla quale caricate l'immagine in fase di editing deve essere la stessa durante il rendering, altrimenti «TS» visualizzerà un messaggio di errore e proseguirà l'elaborazione senza utilizzare lo stencil.

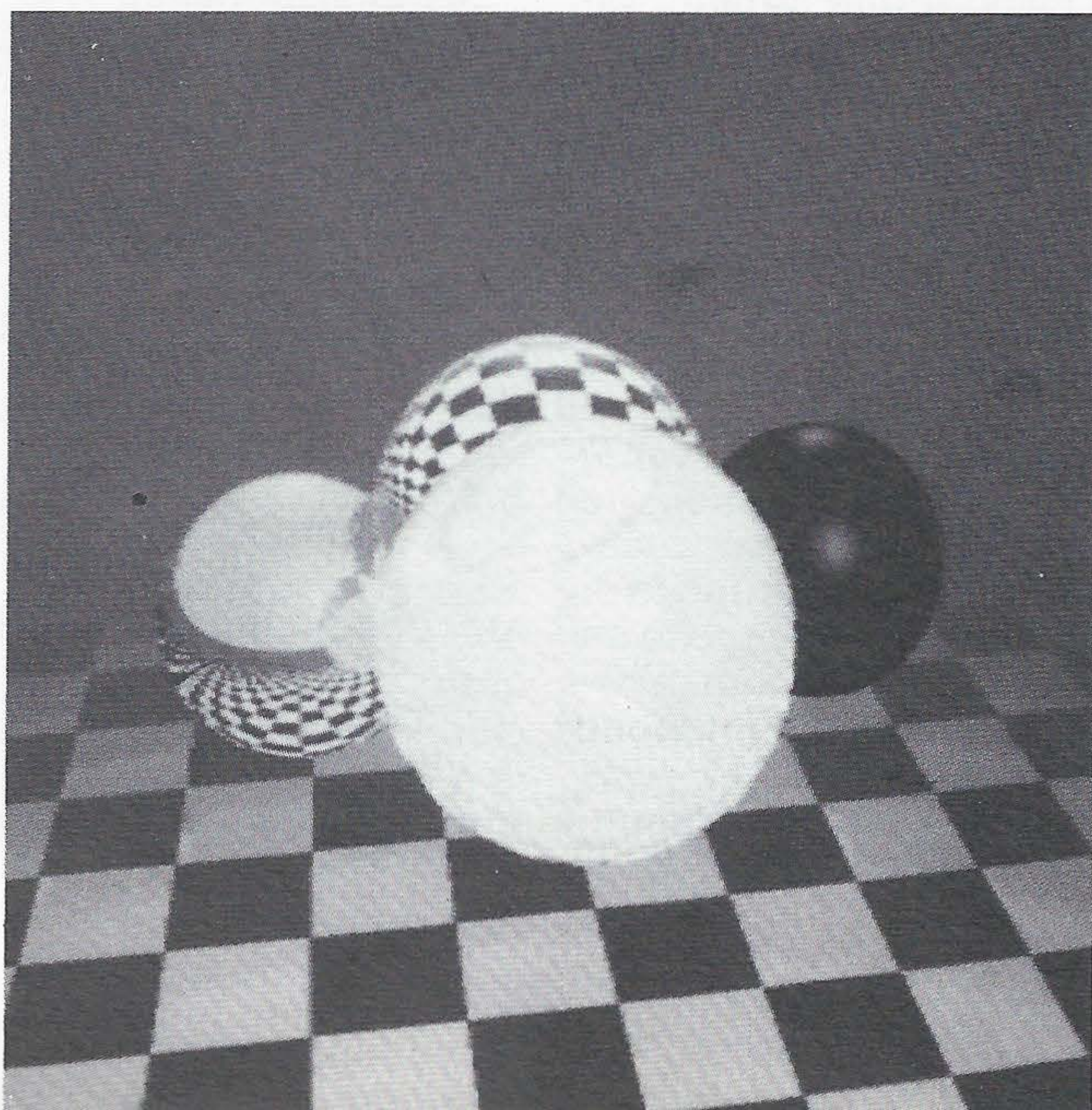
L'USO DEI BRUSH

Brush in inglese significa pennello. I brush sono, per intenderci, quelle porzioni di schermo ritagliate con il classico tool delle forbici sui programmi di grafica pittorica. Salvando su disco un brush, («**DPaint III**» ha un intero menu dedicato alla loro gestione) esso potrà essere tranquillamente caricato su «TS» ed usato come un'etichetta da applicare su di un oggetto a vostra scelta. Le possibilità artistiche offerte dai brush sono pressoché infinite, anche se il loro corretto utilizzo comporta una certa esperienza.

Ricordate che un brush

non può essere caricato direttamente sull'oggetto, perché ovviamente il programma non saprebbe dove incollarlo esattamente; create perciò un asse, selezionatelo, andate su «**Attributes**» e, in basso a destra, scegliete «**Iff Brush**».

Uscite e accedete al **menu «Brush»**, contenente una serie di caselle vuote (potete infatti inserire fino ad otto brush in una scena), selezionate quella che preferite, ed attivate il «**Load**».

Caricate il brush che avrete preparato precedentemente; come al solito, la sua posizione sul disco non dovrà variare quando passerete alla fase del rendering, o il brush non potrà essere utilizzato. Le dimensioni dell'asse al quale il brush è stato associato saranno anche quelle dell'etichetta alla fine del rendering: cambiatele quindi a vostro piacimento, ricordando che aumentandole troppo rischiate di «sgranare» l'immagine (come quando fate uno zoom in un programma di grafica pittorica).

Adesso posizionate l'asse a poca distanza dall'oggetto, nel punto in cui vorreste applicare l'etichetta, come se doveste applicare un adesivo, e prendete ad occhio le misure per metterlo più dritto possibile. Raggruppate ora l'oggetto e l'asse: selezionate l'asse (**per primo!**) e, tenendo premuto **Shift**, selezionate l'oggetto e date il comando «**Group**».

Il brush è stato ora applicato, e dovete scegliere la modalità di **wrapping**, un comando piuttosto difficile da spiegare. Sotto al **menu «Brush»** trovate un **sottomenu «Wrap»**: con «**Wrap**» potrete decidere se, rispetto ad uno degli assi, l'etichetta deve avvolgere l'oggetto, seguirne la superficie, o rimanere piuttosto contenuta, ben delimitata. «**Flat**» significa piatta, quindi contenuta; «**Wrap**» significa arrotolata, avvolgente.

Gli unici assi sui quali è possibile usare questo effetto sono X e Z, cioè quelli del «**Front View**»: è consigliabile quindi portare a termine questo tipo di operazione con l'oggetto in posizione iniziale e metterlo in posizione per la scena solo successivamente, facendolo ruotare insieme al brush al quale l'avete unito.

Ricordate che l'oggetto al quale attaccate il brush non deve avere un **blending** troppo alto, o il brush si fonderà con il suo colore di superficie.

I brush prevedono anche due eccezioni, cioè due casi in cui il loro comportamento non è quello usuale. Un caso è già stato illustrato sui fascicoli precedenti, ed è l'applicazione di un brush su di un asse usato come fonte di luce: l'asse proietterà il brush su tutti gli oggetti creando effetti molto interessanti.

L'altro caso è quello del «**ground**»: quando applicate un brush al pavimento, esso verrà ripetuto all'infinito come se il brush fosse una piastrella. Anche in questo caso, i risultati possono essere spettacolari.

LE ANIMAZIONI AL MEGLIO

In fatto di animazioni, «TS» non offre il meglio di sé: il suo metodo di animazione è piuttosto scomodo

al punto che, a nostro parere, senza un hard disk è virtualmente impossibile raggiungere i livelli ottenibili con programmi come «Sculpt». Qualcosa di buono però lo si può fare lo stesso: vediamo come.

Prendiamo in esame innanzitutto l'editing dell'animazione: di regola la cella in cui si sviluppa lo schema dell'animazione è la **«Key Cell»**, quella che potete vedere in basso nello schermo iniziale. La «Key Cell» in «TS» viene usata solo nelle animazioni, tanto che non può neanche essere «renderizzata» come una cella normale.

Il fulcro dell'intero sistema di animazione di «TS» è il **requester dello «Story»** che vi abbiamo già illustrato a proposito della funzione «Extrude by path». Dal requester «Story» potete assegnare ad un oggetto un percorso da seguire (con lo stesso sistema usato per l'estrusione), una rotazione, un cambiamento di dimensioni in scala.

È possibile anche fare in modo che, se in una serie di oggetti raggruppati si sta lavorando su quello principale (cioè quello selezionato per primo), tutti gli oggetti «figli», come vengono definiti, lo seguano a ruota selezionando **«Follow Me»**.

Nel momento in cui avrete assegnato ad ogni oggetto un proprio movimento, uscite dalla «Key Cell» e decidete quanti fotogrammi deve avere la vostra animazione: se, ad esempio, per un oggetto avete deciso una rotazione di 360 gradi e scegliete di suddividerla in 10 fotogrammi, l'oggetto ruoterà per ognuno di essi di 36 gradi. Selezionate a questo punto, tenendo premuto il tasto **Shift**, tante celle (vuote) quanti sono i fotogrammi. Date poi il comando **«Make»** dal **menu «Cells»**: «TS» aprirà sul disco tanti file quante sono le celle, preparandosi a riempirli.

Dallo stesso menu selezionate adesso **«Use Story»**, che dirà al programma di tener conto dei movimenti che avete assegnato a ciascun oggetto. Riselezionate la **«Key cell»** e, tramite il solito menu, attivate il comando **«Source»**.

Selezionate infine le celle che avete attivato (sempre con il tasto **Shift**) e date il comando **«Target»**: «TS» inizierà così a preparare ogni cella, variando le caratteristiche degli oggetti passo per passo. Il procedimento è abbastanza lungo perché deve salvare ogni cella con tutti i suoi oggetti separatamente sul disco, senza contare che lo spazio occupato da ogni singola cella non è poco, e moltiplicarlo per dieci o per venti può ridurre notevolmente lo spazio sul disco.

RISPARMIARE SPAZIO

In questo caso «TS» offre una soluzione piuttosto valida: tornate nella **«Key cell»** e salvate ogni oggetto singolarmente sul disco, selezionandolo e utilizzando il comando **«Save»** del **menu «Special»**. Mantenendo sempre selezionato l'oggetto, usate il comando **«Delete»** dal **menu «Edit»**: l'oggetto verrà letteralmente cancellato dallo schermo.

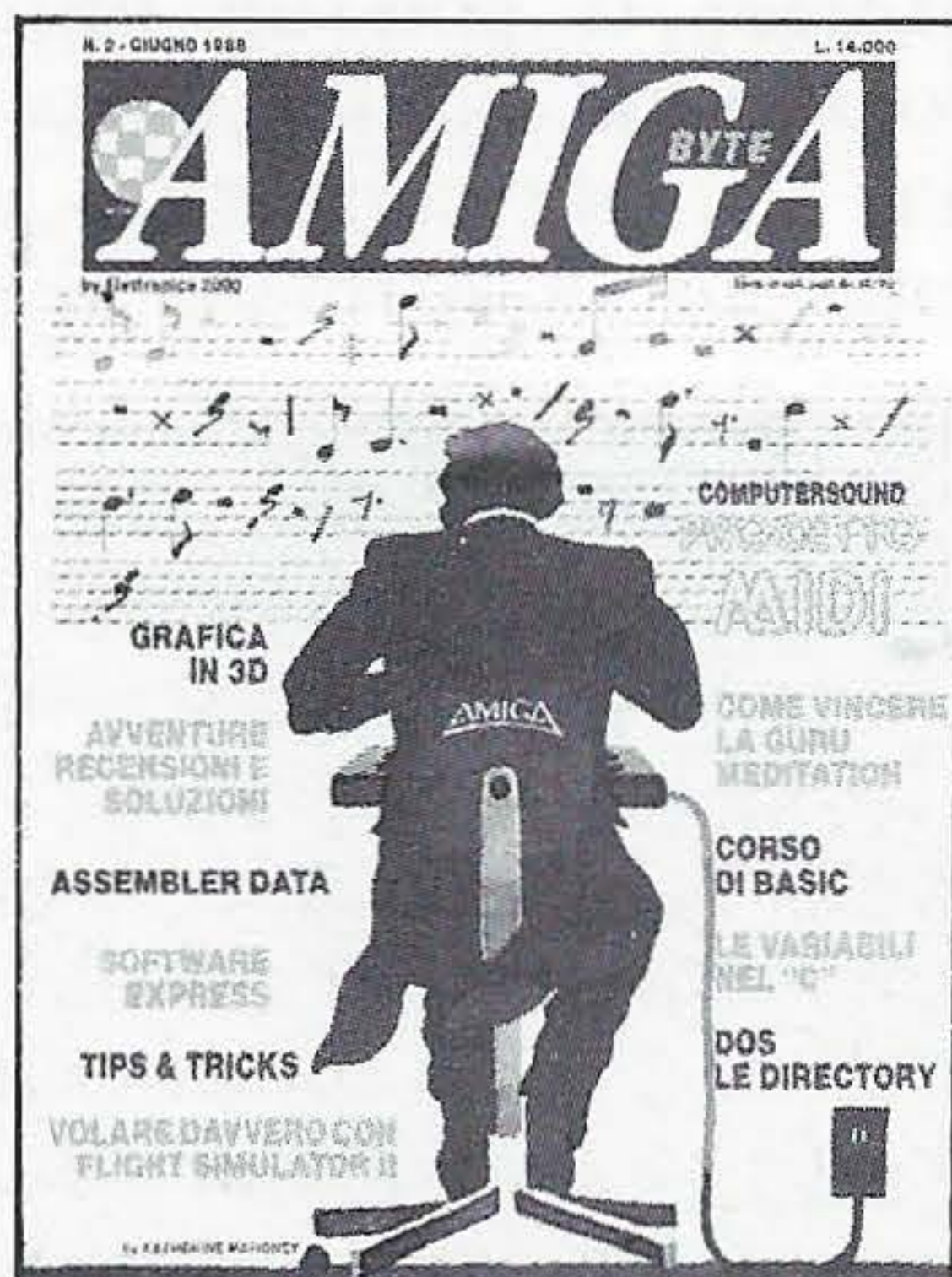
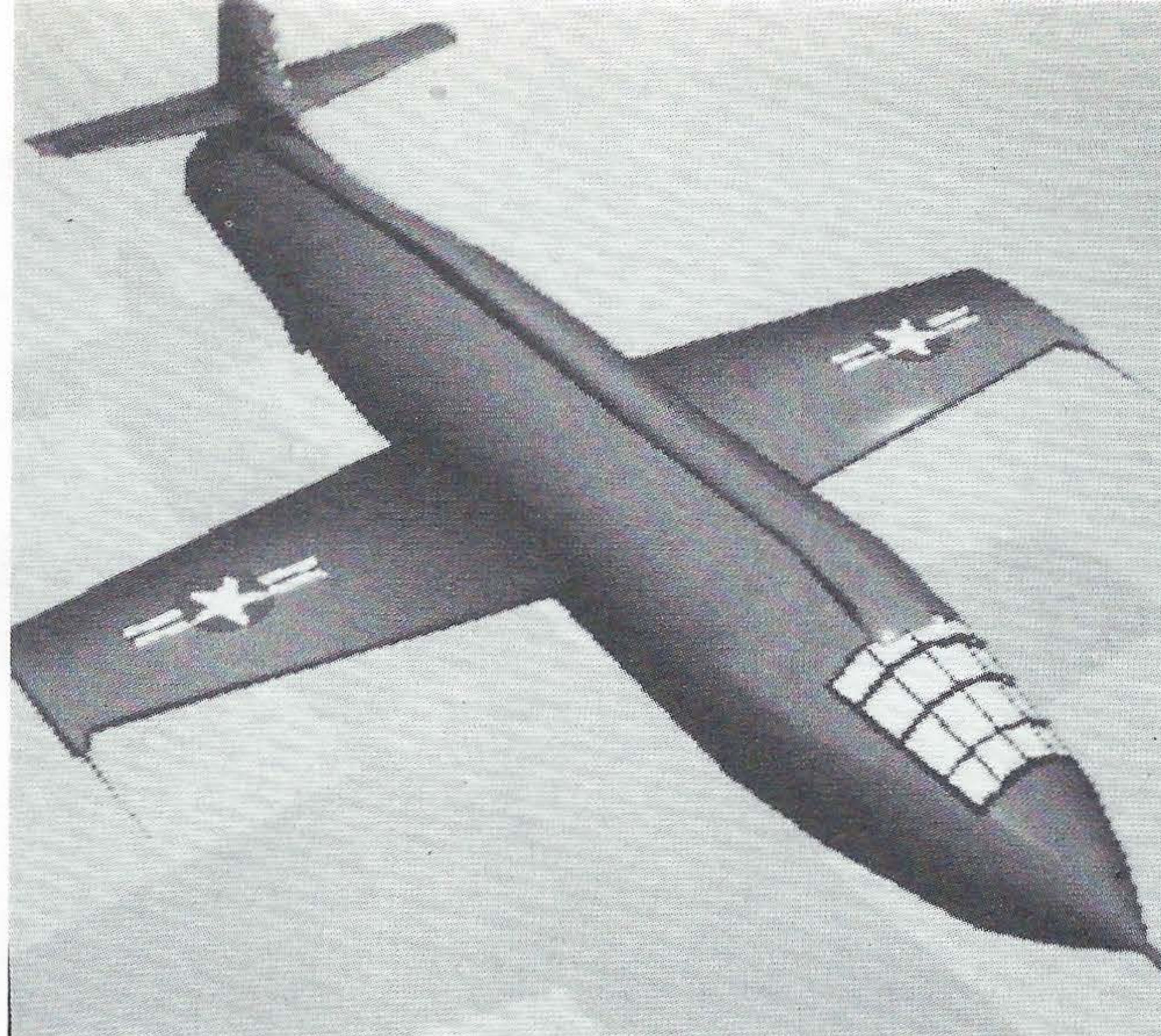
Ora, sempre tramite il **menu «Special»**, date il comando **«External»** e ricaricate con esso l'oggetto.

Esso riapparirà mantenendo tutte le caratteristiche che gli avevate attribuito: se ora uscite dalla **«Key cell»** e ripetete la sequenza precedente, «TS» non salverà per ogni cella tutti gli oggetti, ma solamente i loro nomi e le caratteristiche attribuite (posizione ed orientamento), facendovi risparmiare un'enormità di spazio.

Completato il lavoro di preparazione delle celle, potete scegliere fra tre possibilità: la prima consiste nel rendering delle immagini una per una, esportandole e compattandole con **«Movie»** o con un altro programma simile (ricordate che per convertire le immagini in formato IFF standard dovete usare il tasto «S» dopo aver caricato l'immagine).

Un'altra possibilità consiste nel selezionare tutte le celle da renderizzare e dare il comando **«Make Scene»** dal **menu «Scene»**: in

PHOTO CREDIT

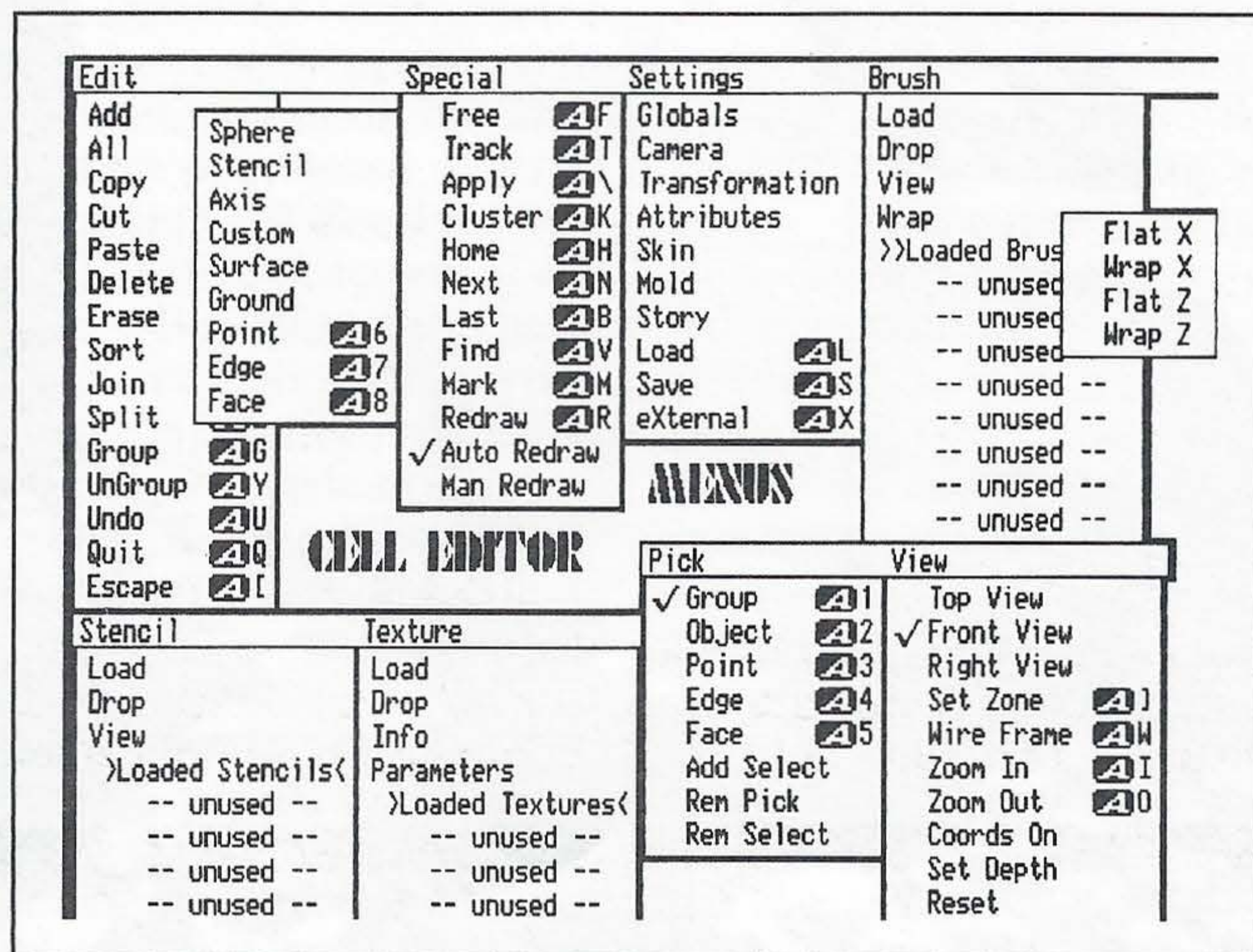
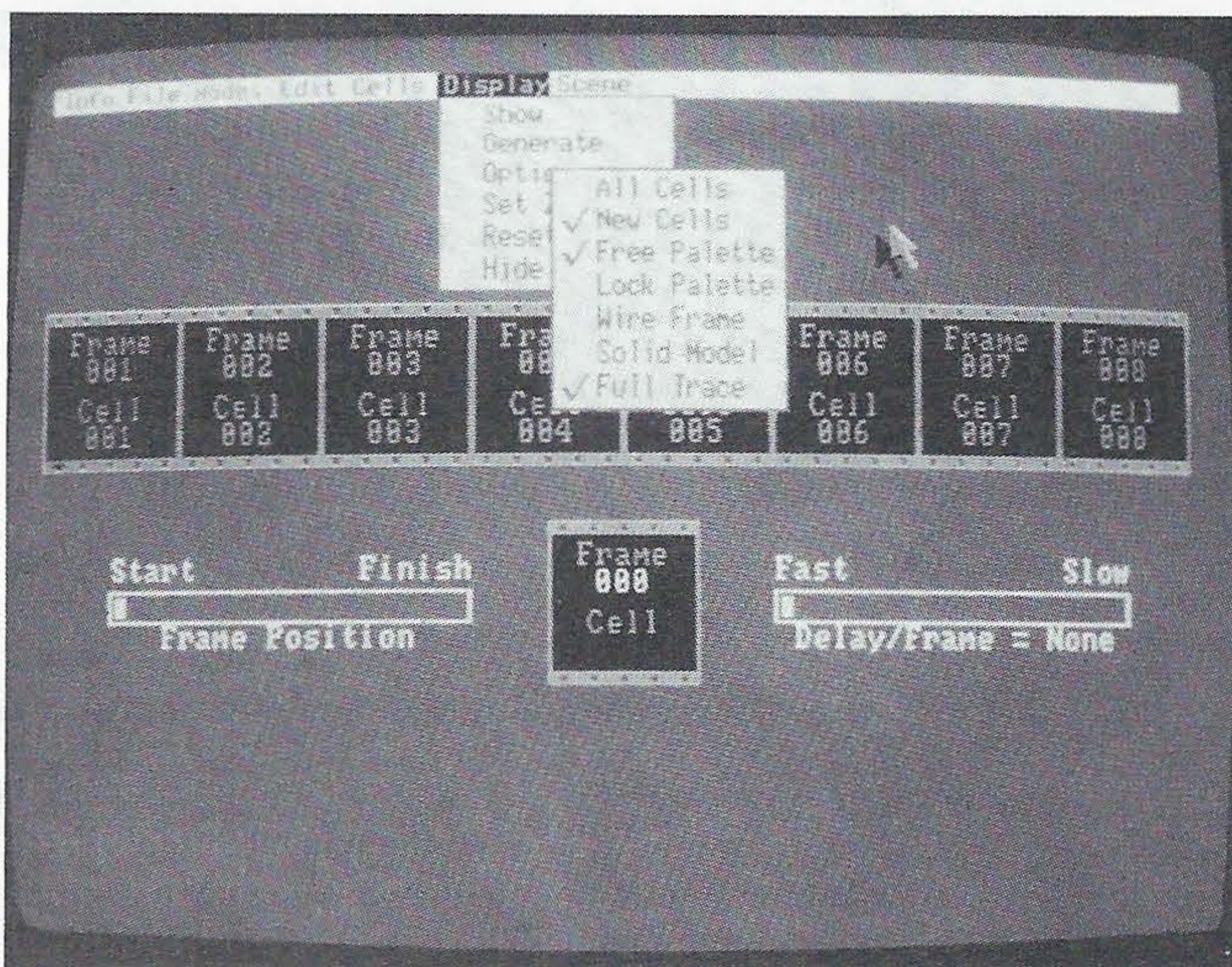
Le immagini che corredano questo articolo sono state create con TurboSilver da Antonio Di Lorenzo di Roma cui vanno i complimenti e i ringraziamenti della Redazione.

questo caso «TS» genererà ogni immagine, compattandola alla fine. Attenzione, poiché il file finale di solito ha dimensioni notevoli e potreste avere molti problemi se lavorate solo con i dischetti!

La terza possibilità è disponibile solo sull'ultimissima versione di «TS», la **3.01SV PRO**, ed è chiamata «**Make Anim**»: essa consiste nella creazione di un file «anim» dopo il rendering di ogni singola immagine. Alla fine otterrete solo un'animazione completa, ma non i singoli fotogrammi che la compongono, risparmiando molto spazio sul disco.

Non conviene invece usare l'opzione «**Make Movie**», probabilmente familiare agli utenti abituali di «Sculpt». «Make Movie» prepara un'animazione identica a quella di «Make Scene», con l'aggiunta di uno Script File che permette di eseguire un vero e proprio montaggio; anche se l'idea è valida, l'utilizzo di questa funzione è molto complesso e richiede una ricca dotazione hardware.

Occupiamoci infine dei parametri di generazione e dei settaggi necessari nel caso di un'animazione: andate al **menu «Display»** e guardate il **sottomenu «Options»**. Innanzitutto, se avete già renderizzato delle scene di prova tra le celle dell'animazione, potete selezionare «**New Cells**» per procedere al rendering solo di quelle rimaste, mentre «**All Cells**» lavorerà su tutte, senza distinzione.

Poi, nel caso di animazioni, vi conviene selezionare «**Lock Palette**» perché il fatto che «TS» sia libero («Free Palette») nello scegliere i colori da cella a cella può portare a pessimi effetti nell'animazione completa.

Tra i tre modi di rendering, infine, avete piena libertà di scelta: potete optare per «**Wire Frame**» (che funziona solo con le animazioni) per avere un veloce preview; oppure selezionare «**Solid Model**» (che accelera notevolmente i tempi di rendering, ma può causare occasionalmente degli errori); o scegliere «**Full Trace**», per usare il quale è bene disporre di un Amiga 3000 o comunque di una scheda acceleratrice, visto che la generazione di ogni singola immagine in tale modalità può richiedere anche parecchie ore.

L'immagine mostra tutte le opzioni dei menu del Cell Editor. In basso, la schermata principale del programma.

HiSoft Basic

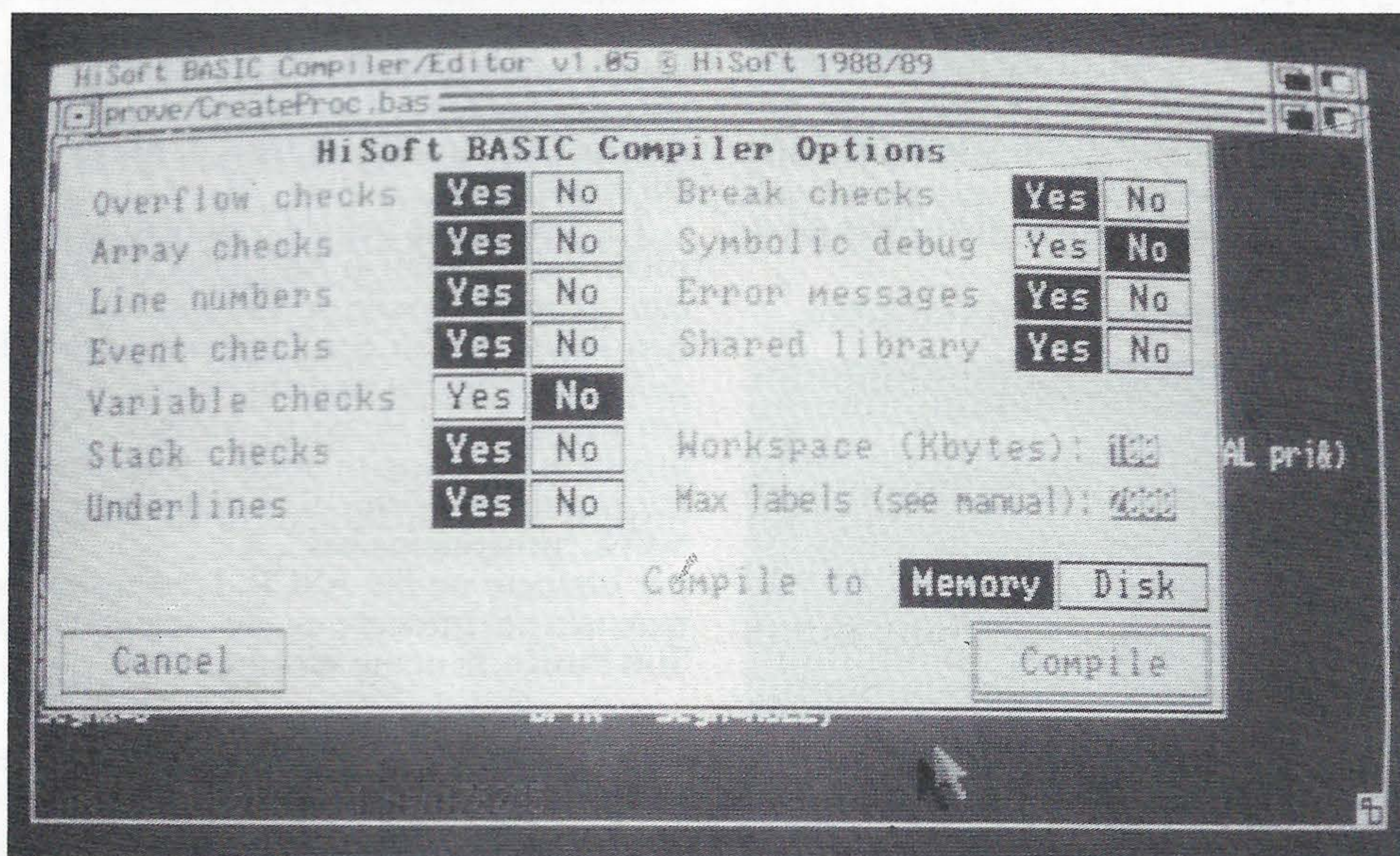
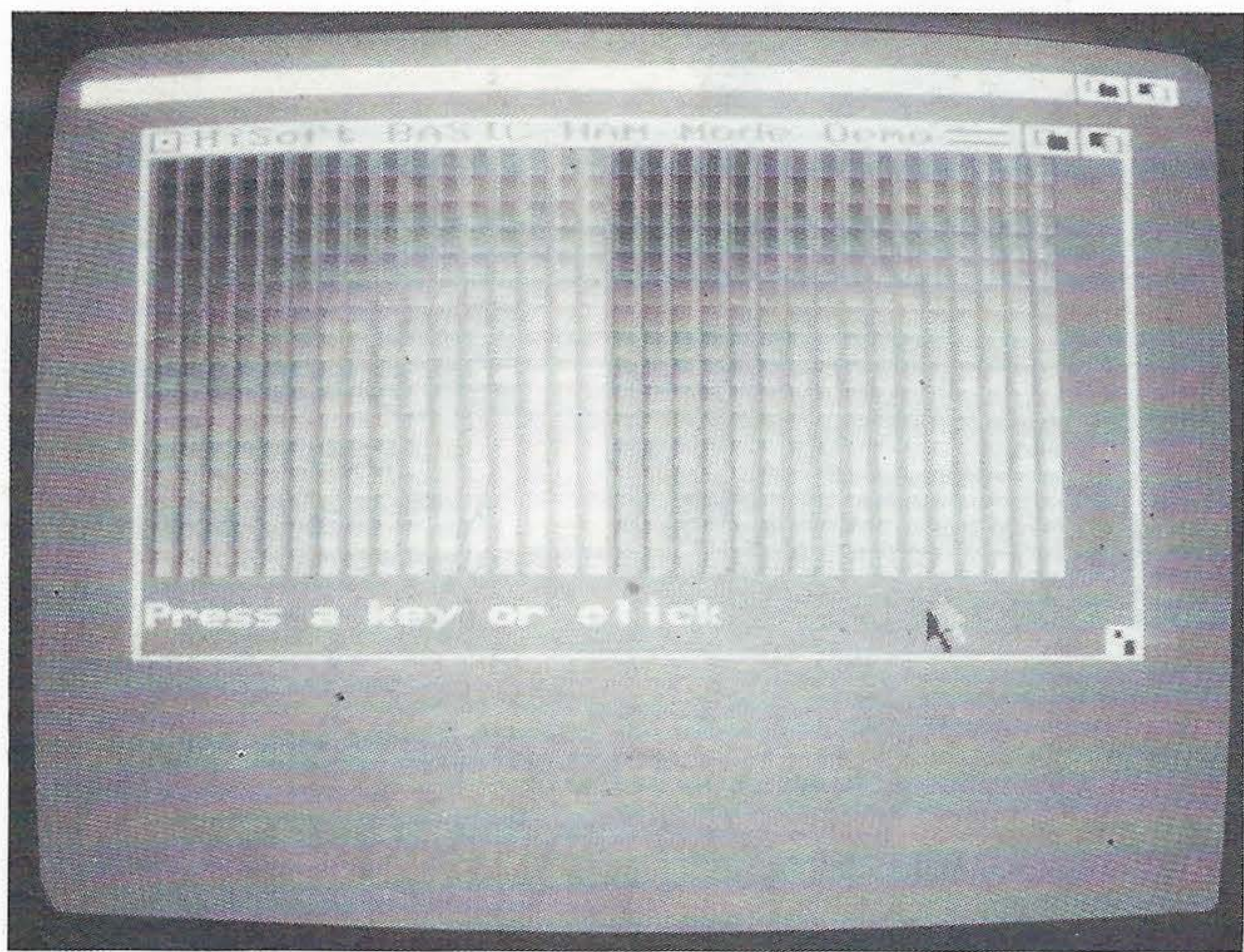
Un compilatore per chi vuole produrre codice efficente e veloce con il minimo sforzo, sfruttando un linguaggio semplice, compatibile con AmigaBasic, ed un'interfaccia intuitiva.

di PAOLO BOZZO

Facilità, potenza, compatibilità: sono queste le qualità che in genere più si richiedono ad un linguaggio di alto livello, ed è in questa direzione che si sono mossi i programmatori della HiSoft cercando, con «HiSoft Basic», di raggiungere il compromesso ideale tra le esigenze, spesso contrastanti, di utenti e programmatori. Compatibilità elevata con AmigaBASIC (oltre che con il QuickBasic del mondo MS-DOS), velocità d'esecuzione, possibilità di utilizzare la programmazione strutturata e di effettuare il *linking* con i classici *ALink* e *BLink*, interfaccia intuitiva e praticità d'uso sono caratteristiche che solo in «HiSoft Basic» troviamo unite insieme.

UN AMBIENTE INTEGRATO

La confezione, con i suoi due dischi, si presenta in modo piuttosto elegante. Purtroppo, manuale e programma sono in inglese: c'è solo da attendere che l'occhio attento della Lago o della Leader Distribuzione cada sul programma e che venga prodotta una versione in italiano di tutto il pacchetto, così come è avvenuto per l'*assembler* DevPac della stessa HiSoft. Il manuale, che consta di oltre 360 pagine, appare comunque ben strutturato: segue ad una veloce introduzione iniziale un intero capitolo, il secondo, dedicato agli esempi classici per i non esperti (una rubrica telefonica e le ben note «torri di Hanoi»); ecco poi le istruzioni sull'uso dell'*editor* ed i concetti fondamentali di questa implementazione BASIC (capitoli terzo e quarto); il quinto capitolo contiene invece la descrizione in ordine alfabetico di tutti comandi, piuttosto sintetica ma efficace ed indispensabile per una consultazione veloce; infine, una serie di appendici ci informa, a volte in modo un po' troppo affrettato, su numerose particolarità del compilatore.

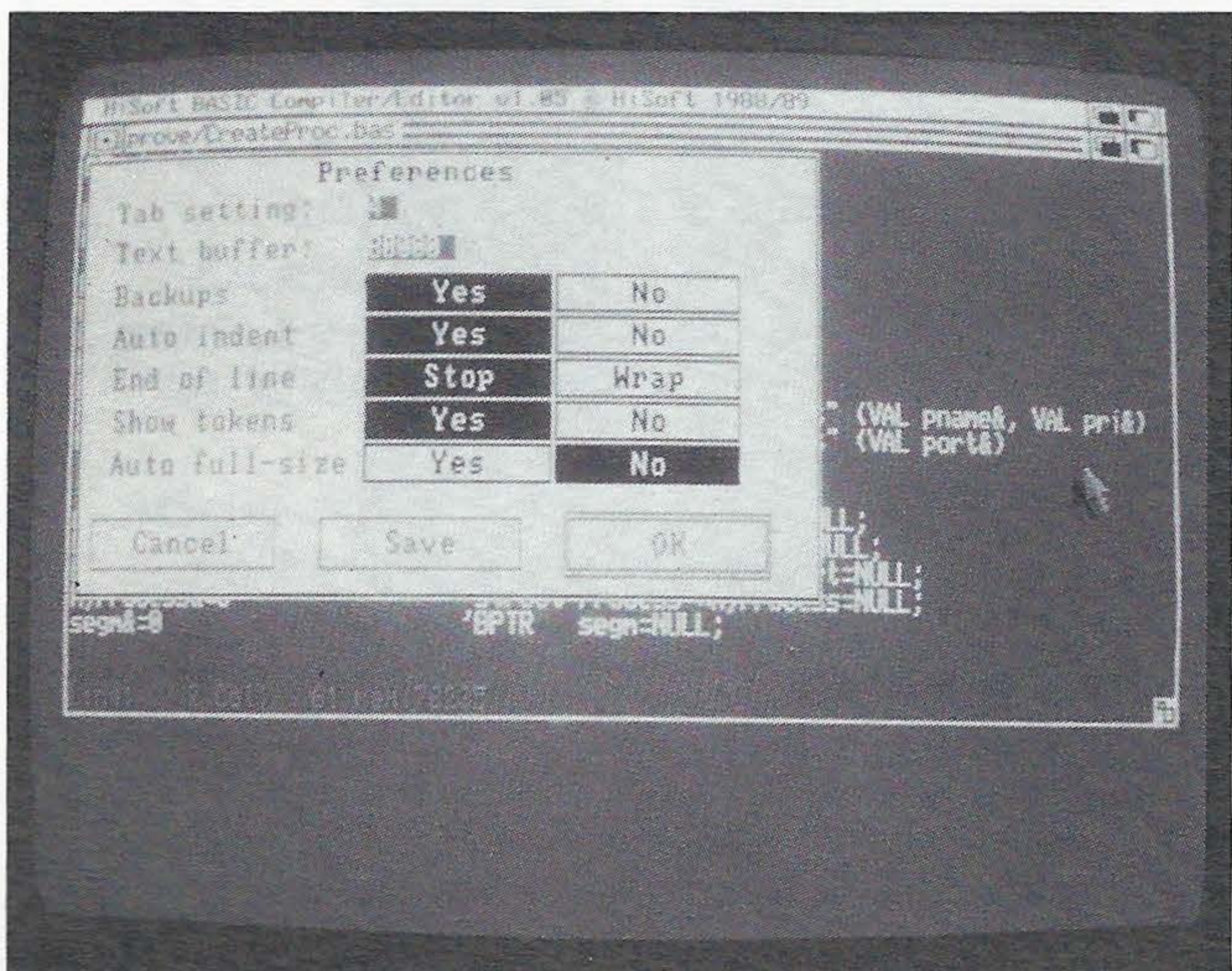
Il pacchetto consta essenzialmente di un *Text-Editor* («HiSoft BASIC») e del compilatore vero e proprio («hb.compiler»). I due strumenti (*tools*) che ci vengono forniti risultano ben integrati; dall'interno del *Text-Editor* abbiamo infatti la possibilità di richiamare anche il compilatore semplicemente con un'opzione del menu

Quadro di richiesta per l'utilizzo del compilatore dall'interno del Text-Editor.

HiSoft BASIC permette di gestire facilmente i 4096 colori del modo HAM.

(«compile», ovviamente) e di scegliere le opzioni di compilazione tramite un comodo quadro stile-*intuition*. Il tutto è stato ricalcato dal già collaudato ed efficiente ambiente del «DevPac assembler» della stessa HiSoft. Anche qui (come in «DevPac») abbiamo la possibilità di scegliere fra la compilazione su disco, per salvare il file eseguibile, e quella in memoria, per poter testare immediatamente il programma, avvicinandoci così molto alla praticità di un interprete; tra l'altro, se la compilazione fallisce, l'elenco degli errori rimane in memoria e possiamo comodamente portarci su di essi premendo contemporaneamente i tasti *amiga-destro* e *J* (opzione «jump to error» del menu «program»).

Sia l'*editor* che il compilatore sono utilizzabili con facilità da CLI come da WorkBench, tuttavia nelle nostre prove abbiamo trovato più comodo lanciare l'*editor* con compilatore integrato da WorkBench, riservando il CLI per l'uso del solo compilatore, specialmente se si produceva un file oggetto che avrebbe poi dovuto essere *linkato* (seguendo così lo stile «austero» di molti compilatori C e Modula 2).

Un particolare occhio di riguardo è stato tenuto per i possessori di Amiga 500 con soli 512 K di ram: se siete uno di questi, dovrete usare *editor* e compilatore contenuti nel primo disco insieme ai file di sistema, altrimenti potrete utilizzare i programmi corrispondenti ma più completi che si trovano nel secondo disco insieme a numerosi esempi.

L'*editor* è praticamente lo stesso del «DevPac» ed è,

Il quadro di richiesta per le preferences del Text-Editor.

fortunatamente, molto più efficiente e completo di quello dell'AmigaBASIC. È particolarmente utile la possibilità di scegliere e salvare, tramite un apposito quadro di richiesta, le nostre preferenze personali, tra cui la conversione automatica delle parole-chiave in maiuscolo (**show tokens**), il rientro automatico (**auto indent**), la dimensione del **buffer** di testo, il numero di caratteri del tabulatore (**tab setting**) o l'utilizzo di una finestra in dimensioni PAL (**auto full-size**). È sempre pratico e comodo anche il **File-Request** tratto dalla arp.library (ricordate di copiarla sul vostro disco di lavoro!).

Una nota negativa consiste invece nei numerosi comandi implementati nello stile, ormai veramente obsoleto, di «**WordStar**» (basti un esempio: per cancellare una linea dobbiamo premere contemporaneamente *CTRL* e *Y*, senza aver la minima possibilità di effettuare l'operazione da menu). Macchinose e poco pratiche risultano pure tutte le operazioni su blocchi di testo, che necessitano di particolari combinazioni dei tasti funzione, piuttosto difficili da ricordare, con l'unica consolazione che, premendo il tasto *HELP*, appare una sorta di pro-memoria in tal senso, evitandoci, almeno, di consultare conti-

```
esempio_1
' Semplice esempio di utilizzo del puntatore SYSTAB
' e delle librerie di sistema...
' La chiamata a WaitPort() ci fa attendere sulla porta-
' messaggi di Intuition. Corrisponde al comando SLEEP.

CONST INTUIPORT=4   ' (costante per maggiore leggibilità)

LIBRARY "exec"   ' (può essere evitata l'estensione .library)
LIBRARY "dos"

WaitPort&(PEEKL(SYSTAB+INTUIPORT))

PRINT "evento intuition: ";
IF MOUSE(0)<>0 THEN
    PRINT "mouse sinistro."
ELSEIF INKEY$<>"" THEN
    PRINT "tastiera."
ELSE
    PRINT "ignoto."
END IF

Delay&(150)       ' (attendi tre secondi)
LIBRARY CLOSE
SYSTEM
```

Esempio 1

nuamente il manuale. Una volta accettati questi limiti però, ben raramente sentiremo la necessità di ricorrere al nostro *editor* preferito rinunciando alla possibilità di gestire direttamente il compilatore e gli errori (oppure affidandoci alla titanica impresa di riscrivere tutta l'interfaccia con ARexx).

Un'avvertenza per chi è abituato a lavorare con l'interprete: i programmi in AmigaBASIC possono essere caricati e compilati solo se sono in formato ASCII, quindi non vanno salvati con l'opzione **save** del menu **project** dell'interprete, ma bisogna ricorrere al comando diretto; per esempio, se siamo in AmigaBASIC ed abbiamo in memoria un listato di nome «programma.bas», dovremo *clickare* nella finestra di output e battere:

SAVE "programma.bas", A

per ottenere un file ASCII. L'estensione «.bas» è raccomandata per ottenere un file eseguibile, generato da «HiSoft Basic», di nome «programma».

LA LIBRERIA CONDIVISA

Il codice prodotto dal compilatore è abbastanza compatto e generalmente piuttosto efficiente. Di norma i compilatori BASIC abbisognano di una vasta libreria di

L'help illustra le possibili operazioni (copia, spostamento, ecc.) sui blocchi di testo.

funzioni per implementare tutte le facilitazioni offerte dal linguaggio; la gestione di questa libreria è affidata alla fantasia ed alla tecnica dei programmatori. Nel nostro caso si ricorre alla soluzione più logica e razionale per Amiga, la libreria condivisa (**hisoftbasic.library**), che viene caricata in memoria una volta per tutte ed alla quale possono accedere diversi programmi contemporaneamente. Unico limite di questa impostazione è la rilevante quantità di memoria richiesta da tale libreria, sia nella RAM che su disco, limite superabile specificando l'opzione «L−», che fa in modo che il compilatore inserisca nel file eseguibile soltanto quella parte di codice della libreria necessaria al funzionamento del nostro programma. Questa opzione è particolarmente utile se sul disco di nostra creazione esiste un solo programma compilato con «HiSoft Basic»; in questo caso risparmieremo globalmente molta memoria poiché, per girare, il nostro programma non avrà bisogno della libreria e sarà così indipendente da file esterni, salvo che non si usino operazioni con la virgola mobile, per le quali sono comunque necessarie le librerie matematiche standard.

Se invece intendiamo produrre qualche decina di piccoli programmi, ovviamente sarà più vantaggioso utilizzare la libreria tramite l'opzione «L+».

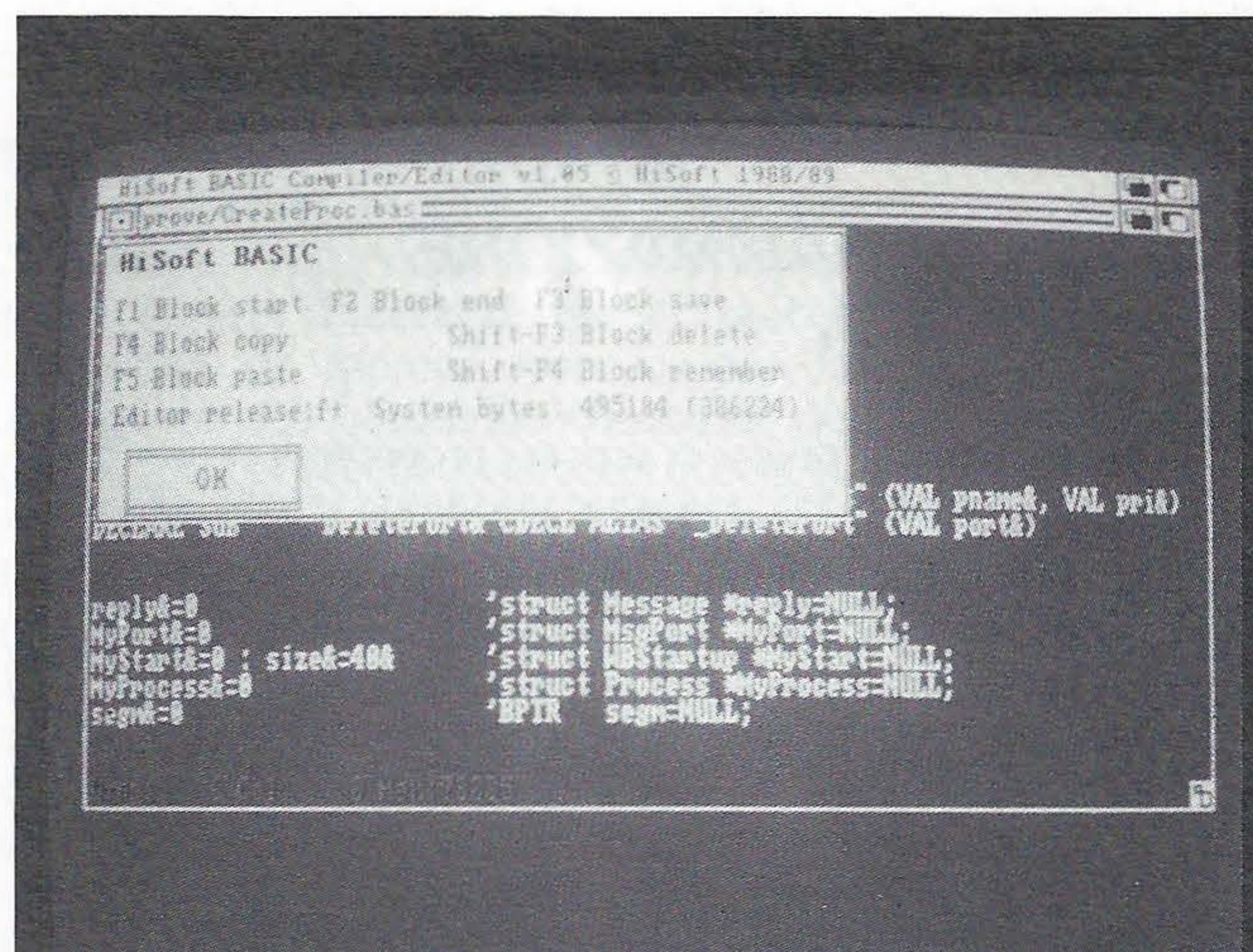
```
esempio_2
' esempio di chiamata di funzioni in linguaggio C (da amiga.lib).
' CDECL serve per dichiarare al compilatore che è in linguaggio C-assembler
' ALIAS serve per comunicare al compilatore il nome esatto della routine
' sono necessari BLink (directory c:) e amiga.lib (directory lib:).
' E' richiesto l'ambiente CLI. Compilare così:
' 1> hb.compiler esempio_2
' 1> BLink esempio_2.o TO esempio LIB lib:amiga.lib DEFINE _SysBase 4 ND
' provare digitando "esempio".

REM $option y,g-

DECLARE FUNCTION CreatePort& CDECL ALIAS "_CreatePort" (VAL pname&, VAL pri&)
DECLARE SUB      DeletePort  CDECL ALIAS "_DeletePort" (VAL port&)

MyPort&=CreatePort&(0,0)
IF MyPort&=0 THEN
    PRINT "Niente porto di comunicazione!"
    END
ELSE
    PRINT "Creato porto di comunicazione"
END IF

CALL DeletePort(MyPort&)
END
```

Esempio 2

Facendo comunque un confronto tra «HiSoft Basic» ed il suo diretto concorrente («AC-Basic» della AbSoft), possiamo notare che la lunghezza del file eseguibile è sempre vantaggiosa per l'«HiSoft» e che solo i file prodotti da quest'ultimo possono essere compattati con appositi programmi (come «**PowerPacker**»).

Anche per quel che riguarda la velocità di esecuzione dei programmi, abbiamo riscontrato prestazioni sempre leggermente superiori ad «AC-Basic», nemmeno lontanamente paragonabili a quelle di AmigaBASIC; si cede solo qualche punto al nuovo compilatore «GFA Basic», che tuttavia non è compatibile con AmigaBASIC.

I METACOMANDI

Caratteristica avanzata di questo compilatore è la possibilità di utilizzare particolari **metacomandi**, vale a dire istruzioni su come deve essere effettuata la compilazione (equivalenti, in linguaggio C, ai comandi del preprocessore marcati con #). I metacomandi vanno inseriti dopo la parola chiave **REM** (affinché non impediscano al programma di girare con l'interprete) e devono essere preceduti dal simbolo $. Per esempio:

REM $EVENT OFF

disabilita la gestione di eventi-intuition (mouse, tastiera) e rende più compatto e veloce, a partire da quel punto, il programma. Ovviamente.

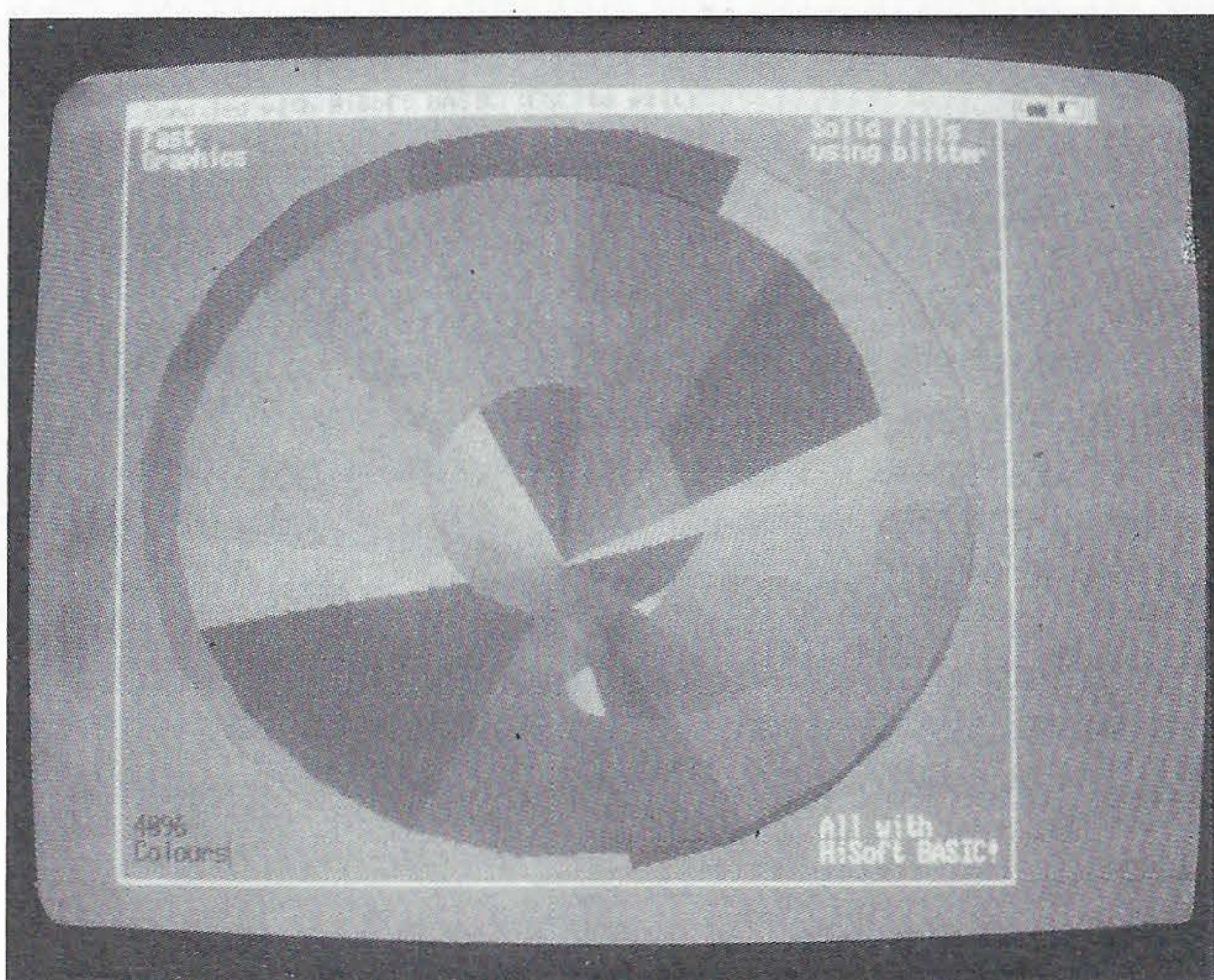
REM $EVENT ON

riporta allo stato «normale» di gestione degli eventi. Il metacomando **$INCLUDE** («#include» del linguaggio C) permette di includere nel nostro sorgente un altro file, senza doverlo inserire manualmente. **$OPTION**, infine, serve per definire la lista delle opzioni di compilazione.

Queste opzioni consistono in una lettera seguita dal segno + oppure − (segni che equivalgono ad un *sì/no* o ad un *on/off*) e vanno separate dall'operatore virgola. Oltre che con il metacomando *$option* (che rimane comunque il sistema più efficace ed universale) queste opzioni possono essere anche scelte, sia pur con alcune limitazioni, tramite i comodi *gadget* del quadro di compilazione.

Un terzo metodo per definire le opzioni consiste nell'inserirle alla fine della linea del comando CLI con il quale si invoca il compilatore, facendole precedere da un trattino e ricordandosi comunque, nel caso in cui la linea termini con un +, di immettere ancora uno spazio vuoto, per non confondere il CLI.

La scelta offerta tra le varie opzioni è piuttosto ampia e potente e soprattutto, tramite esse, è sovente possibile dare un aspetto più professionale e maggiore efficienza e versatilità al nostro programma.

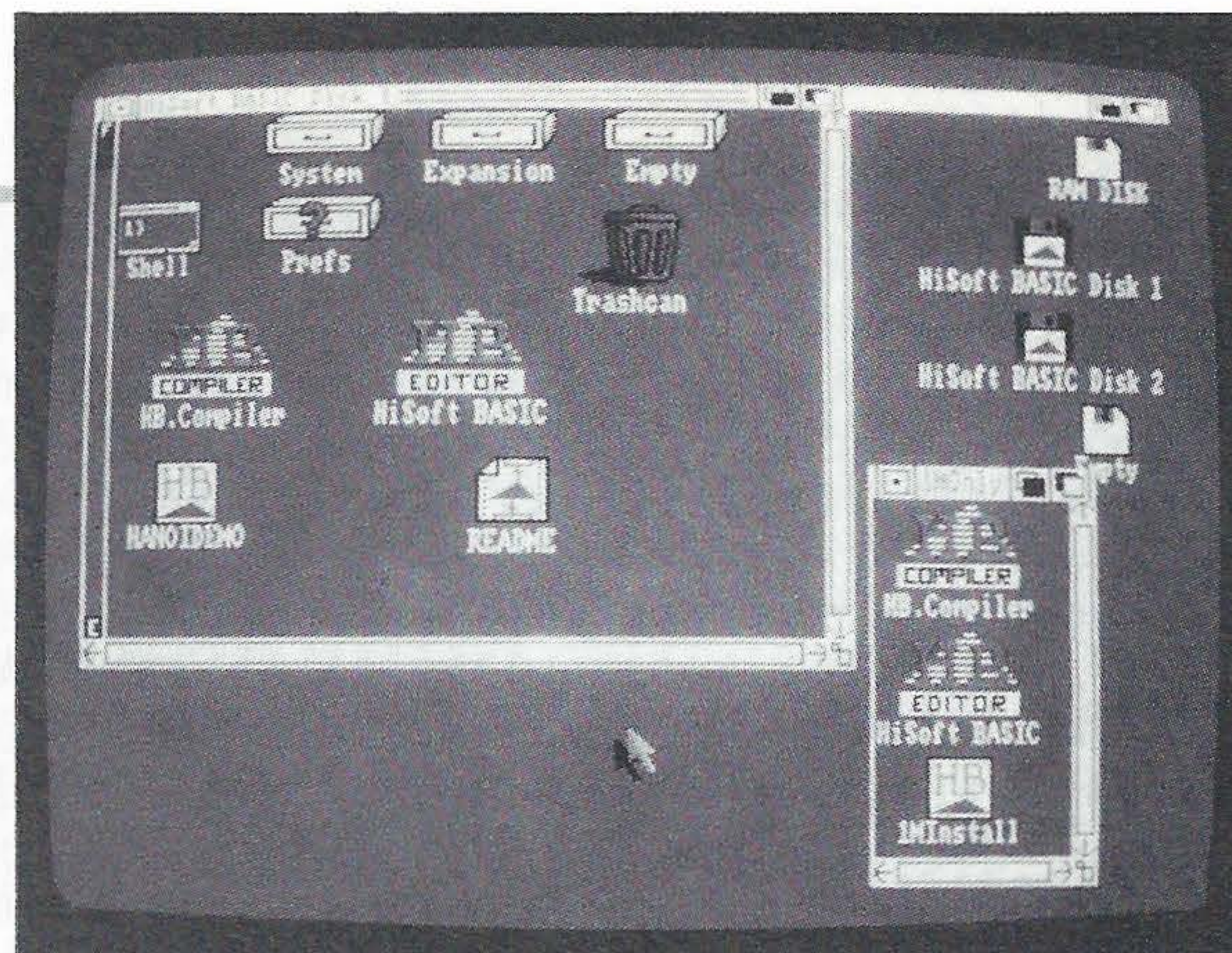
Bell'esempio degli effetti grafici ottenibili con HiSoft BASIC.

LA PROGRAMMAZIONE AVANZATA

«HiSoft Basic» permette di scrivere programmi secondo i principi della programmazione strutturata, grazie all'implementazione di potenti strutture di controllo. Per esemplificare queste possibilità offerte dal pacchetto, immaginiamo di dover realizzare un semplice menu con le opzioni «apri», «salva», «esci». Dopo averle rese esplicite, con:

```
PRINT "MENU"
PRINT "A: Apri"
PRINT "S: Salva"
PRINT "E: Esci"
```

dovremo gestire l'input dell'utente. Possiamo risolvere il nostro problema facendo ricorso al ciclo DO... LOOP:

```
DO
   INPUT "Effettua la tua scelta: ",scelta$
   IF scelta$=="a" OR scelta$=="s"
      PRINT "OK"
   ELSEIF NOT(scelta$=="e") THEN
      PRINT "errore"
   END IF
LOOP UNTIL (scelta$=="e")
```

DO segna l'inizio della struttura di controllo, LOOP UNTIL... (o LOOP WHILE...) la fine. Si esce dal ciclo quando l'espressione dopo UNTIL è vera, altrimenti si ritorna all'istruzione successiva a DO. L'ultima linea avrebbe anche potuto essere scritta così:

```
LOOP WHILE NOT(scelta$=="e")
```

perché WHILE è l'esatto contrario di UNTIL e ci fa uscire dal ciclo solo quando l'espressione che lo segue è falsa. Si sarà anche notato che abbiamo scritto == in luogo di =; si tratta di una forma molto comoda che ci permette di non distinguere tra stringhe con lettere maiuscole e stringhe con lettere minuscole, o di operare con una certa approssimazione su dati numerici.

LE OPZIONI PRINCIPALI

Le principali opzioni permettono di attivare o no: il controllo dello **stack** (**X**) e degli **arrays** (**A**), perché non eccedano il loro dimensionamento; il prelievo del **break**, quando l'utente preme CTRL+C oppure *amiga-destro+punto* (**B**); la comparsa di un chiaro messaggio di errore (per chi conosce l'inglese - lettera **E**); la gestione dei numeri di linea (**N**); la produzione di icone associate ai file (**G**); il controllo degli **overflows**, nel caso in cui un tipo numerico vada oltre la sua capacità (**O**); la verifica del tipo delle variabili presenti nei sotto-programmi o funzioni (**V**); l'utilizzo della libreria condivisa (**L**). L'opzione **S** attiva la generazione di un file eseguibile adatto per il **debug** simbolico con il programma **Monam2** del «DevPac»: è una caratteristica destinata ai più esperti ma molto potente, in quanto *Monam2* è senz'altro tra i migliori debugger-disassemblatori che ci siano in circolazione. Utile è anche l'opzione **Y**, che attiva la soppressione della finestra iniziale, che in molti casi può essere inutile ed antiestetica (se non abbiamo aperto finestre ed il programma è stato lanciato da CLI, viene usata la finestra CLI come canale di INPUT/OUTPUT). L'opzione **K** permette di definire lo spazio di memoria messo a disposizione per i dati del programma (per esempio «K30» alloca 30 *kilobytes*, mentre se non si specifica nulla vengono utilizzati 20 K). Poiché il BASIC ha una gestione delle stringhe piuttosto complessa, anche se non del tutto trasparente per l'utente, esiste poi un'opzione **T**, che permette di variare il numero di **descrittori temporanei**: se facciamo un frequente uso delle stringhe, sarà meglio indicare un valore superiore a 15, che è quello di **default**.

Sempre a proposito di memoria, è possibile variare il numero massimo delle etichette o **label** (**H**) e lo **stack** matematico, utile se facciamo molti calcoli (**M**).

Infine, come nel caso dell'«AC-BASIC», il compilatore si riserva uno spazio di lavoro che, nel caso di listati di una certa mole, può essere insufficiente; poiché questo spazio deve essere allocato PRIMA che il compilatore inizi a leggere il file sorgente, non può essere definito dall'interno di esso tramite *$option*; se si lavora con l'*editor* non rimane che utilizzare il **gadget «Workspace»**; altrimenti, in ambiente CLI, occorre utilizzare la parola-chiave **HEAP**. Ammettiamo di dover compilare da CLI un file sorgente chiamato «programmone.bas» di 100 K, con gestione degli errori e dei *break*, e libreria condivisa; il nostro comando, che dovrà fare i conti con le necessità della memoria, potrà essere più o meno il seguente:

hb.compiler HEAP=300 «programmone» −b+,e+,l+,t30,k40

(N.B.: se non avete almeno un mega di memoria non provateci nemmeno).

CARATTERISTICHE ESCLUSIVE

Un evidente motivo che può spingere a scegliere «HiSoft Basic» è la sua compatibilità con AmigaBasic, compatibilità che ad altri compilatori, per certi versi più potenti (come «Fast Basic» e «GFA Basic»), è del tutto preclusa. Tuttavia, in seguito ad alcune prove, abbiamo notato che il livello di compatibilità è in genere leggermente inferiore a quello di «AC-Basic», anche se a vantaggio di «HiSoft» va precisato che esistono molte caratteristiche e comandi aggiuntivi chè potenziano ed estendono le capacità dell'interprete.

In primo luogo, va detto che è stato fatto un buono sforzo per consentire una programmazione il più possibile strutturata. Lo stesso manuale si scaglia contro il vecchio stile di programmazione Basic, ricco di **GOTO** ed **IF**, che fa somigliare i listati ad una «massa di spaghetti

In luogo di DO... LOOP avremmo anche potuto usare REPEAT:

```
REPEAT my_menu
  INPUT "Effettua la tua scelta: ",scelta$
  scelta$=LCASE$(scelta$)
  SELECT CASE scelta$
  CASE "a","s":
    PRINT "OK"
  CASE "e":
    EXIT my_menu
  CASE REMAINDER:
    PRINT "errore"
  END SELECT
END REPEAT
```

L'inizio del ciclo è segnato da REPEAT più un nome (che non deve essere usato per altri scopi, vale a dire come nome di una variabile vera e propria, ma può ancora essere riutilizzato come nome di un altro ciclo REPEAT), la fine da END REPEAT; REPEAT dà luogo ad un ciclo «infinito»: quando si giunge alla fine (END REPEAT) si torna SEMPRE all'inizio. Si può uscire da questo «circolo vizioso» solo con l'istruzione EXIT seguita dal nome del ciclo (EXIT può essere usato per uscire da qualsiasi ciclo, funzione o sottoprogramma). Questo esempio ci è servito anche per illustrare l'uso di SELECT CASE, che passa in rassegna i vari «casi» in cui si può trovare una variabile di qualsiasi tipo; notate CASE REMAINDER, che in pratica significa «in tutti i casi rimanenti».

appiccicosi e stracotti»; l'esempio del **Pascal**, del **Modula 2**, del **C**, può aiutare a risolvere lo «*spaghetti problem*», se preleviamo da questi linguaggi le strutture di controllo. Così, di fronte ai classici FOR e WHILE, abbiamo diverse e potenti alternative, delle quali parleremo a parte.

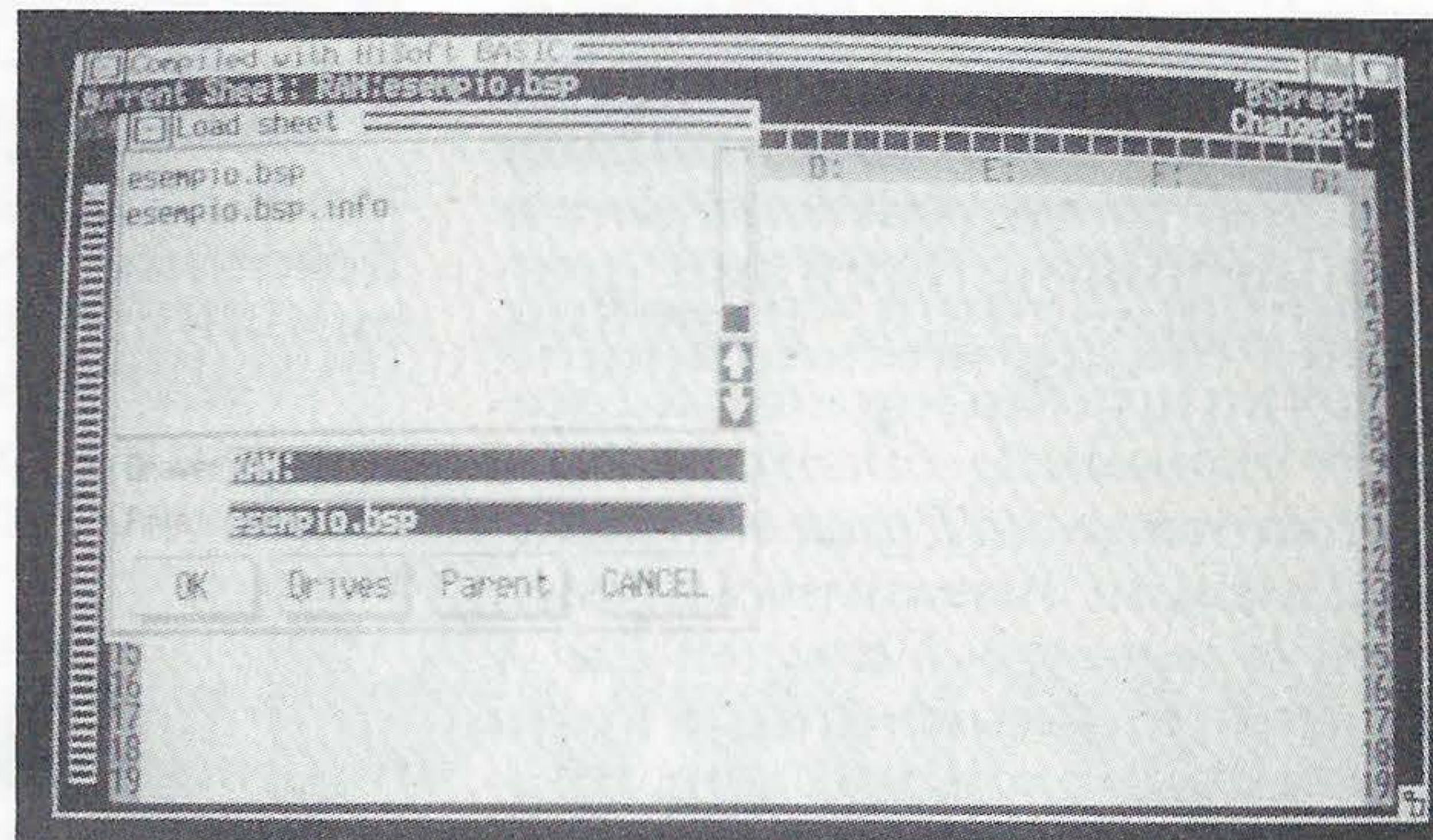
Sono state anche aggiunte varie utili funzioni tra cui i comandi **INCR** e **DECR**, per incrementare e decrementare una variabile; **LCASE$** per convertire una stringa al minuscolo; **FEXISTS** per verificare l'esistenza di un file; **MKDIR** e **RMDIR**, sulla scorta dei Basic del sistema Ms-dos, per la gestione delle directories; **BLOAD** e **BSAVE** per trasferire un file da disco ad un *array* e viceversa; **PCOPY** per la stampa grafica. particolarmente interessanti le due funzioni **COMMAND$** e **SYSTAB**. COMMAND$, se il programma opera in ambiente CLI, restituisce una stringa contenente i parametri con i quali è stato lanciato; le seguenti linee, per esempio:

REM $OPTION Y+,G–
PRINT COMMAND$

potrebbero costituire un rozzo ma funzionante programma «**echo.bas**» (basta ricordarsi di scrivere i parametri senza le virgolette). SYSTAB invece restituisce il puntatore ad una complessa struttura utilizzata dal Basic HiSoft per la gestione dei parametri del sistema e che contiene, fra le altre cose, i puntatori agli schermi usati, alcune indicazioni per la gestione delle icone, il puntatore al *message-port* di Intuition (esempio 1) ed un eventuale puntatore al messaggio di lancio spedito dal WorkBench al nostro programma, ottenibile con **PEEKL (SYSTAB+8)**, che contiene preziose informazioni.

SUPERATI I LIMITI DI AMIGA-BASIC

Infine si è cercato di risolvere il problema delle pesanti limitazioni che AmigaBASIC poneva all'uso delle funzioni e dei sotto-programmi (per inciso ricordiamo che le funzioni ritornano un valore, i sottoprogrammi no); «HiSoft Basic», pur conservando l'impostazione dell'interprete, supera finalmente questi limiti in modo molto più completo di quanto non sappia fare lo stagionato AC-Basic. Le classiche «funzioni definite dall'utente» (**DEF FN**) possono essere scritte su più di una riga. L'istruzione **DECLARE** non è più limitata alle funzioni delle librerie, ma permette anche di dichiarare funzioni (**DECLARE FUNCTION**) o sottoprogrammi (**DECLARE SUB**) scritti in Basic, C od Assembler, consentendo anche di fissare il numero ed il tipo dei parametri e di indicare se vanno passati per valore (preceduti in questo caso dalla parola-chiave **VAL**).

Funzioni e sottoprogrammi possono essere ricorsivi (possono invocare se stessi) ed utilizzare variabili che mantengono il loro valore nelle diverse chiamate (**STATIC**), variabili condivise con il programma principale (**SHARED**) o locali (**LOCAL**), cioè esclusive della routine in questione ed «invisibili» al programma principale ed agli altri sottoprogrammi. Queste possibilità, insieme alla compatibilità con il **linker BLink**, costituiscono l'aspetto più entusiasmante del pacchetto e conferiscono grande flessibilità di programmazione per gli utenti più esperti. La dichiarazione di una funzione in C od Assembler deve seguire una sintassi particolare, in modo da forzare il compilatore a produrre un modulo-oggetto non eseguibile, ma che possa essere unito, tramite il **linker**, ai moduli esterni (esempio 2). Il macchinoso ed ingombrante sistema che aveva l'interprete per chiamare procedure

Abbiamo compilato con HiSoft BASIC uno «spreadsheet» di pubblico dominio (autore B.D. Catley) e vi abbiamo aggiunto il comando «file requester» della libreria ARP.

in assembler, mettendo i dati del modulo-oggetto in un **array**, può tranquillamente essere dimenticato; se proprio vogliamo ancora utilizzarlo, dobbiamo solo ricordarci di sostituire **CALL** con **CALL LOC**. Per gli esperti che volessero utilizzarla, esiste anche la possibilità di ottenere l'indirizzo di un sottoprogramma privo di parametri tramite la nuova funzione **VARPTRS** e di chiamarlo poi con **CALLS** anziché **CALL**; per esempio:

proc&=VARPTRS(Procedura)
CALLS proc&
equivale a:
CALL Procedura

Ovviamente il «trucco» è vantaggioso solo in particolari condizioni e non raccomandabile agli inesperti.

In mezzo a questo «ben di Dio», emergono però alcuni

SEGUE A PAG. 35

Lettere

LA CARTA USO BOLLO

Ho di recente acquistato il programma di pubblico dominio «PrtDrvGen 2.3» allo scopo di modificare il driver per stampante EpsonX/Cbm_Mps-1250 del WorkBench 1.3, per poter stampare su carta uso bollo; ma nonostante i miei sforzi non riesco ad ottenere i risultati voluti.

Valerio Lo Vullo - Caltanissetta

La corretta stampa su carta uso bollo presuppone di poter decidere e variare con precisione la distanza tra le righe di testo. Per intervenire su questa distanza non serve modificare il driver di stampa da te impiegato: se la tua stampante accetta il set di caratteri di controllo Epson, è sufficiente inviare un'apposita sequenza di Escape per impostare la distanza.

Molte stampanti Epson compatibili, ad esempio, riconoscono a questo scopo la sequenza di caratteri Esc A (n), dove (n) è un numero che indica la distanza tra una riga e l'altra, misurata in sessantesimi di pollice (un pollice = 2.54 cm). I caratteri Esc ed A equivalgono, in codici Ascii, ai valori 27 e 65. Perciò se tu, ad esempio, volessi fare in modo che tra le righe di testo sul foglio venga lasciato uno spazio equivalente a mezzo pollice (ovvero a 1,27 centimetri), basterebbe inviare alla stampante i codici ascii 27, 65 e 30. Il valore di interlinea ottimale per le tue esigenze potrà essere trovato facendo qualche esperimento.

Questi codici possono essere inviati in vari modi: ad esempio con un programmino in AmigaBasic da lanciare prima di procedere al caricamento del tuo word-processor preferito.

Conviene comunque che tu consulti il manuale della tua stampante per verificare la corretta sequenza di Escape che abilita la variazione dell'interlinea.

AMIGHI CERCANSI

Sono un abbonato di Amiga Byte e gradirei poter contattare, nelle vicinanze della mia residenza, altre persone che, come me, coltivano l'hobby dell'informatica applicata al computer Amiga.

Sono certo che fra tutti i vostri lettori ed abbonati siano parecchi quelli che risiedono nelle provincie di Siracusa o di Catania.

Vi sarei grato se mi invierete alcuni indirizzi di persone interessate all'Amiga non solo per l'aspetto ludico.

Antonio Artale - Siracusa

Come abbiamo avuto occasione di dirti per telefono, non ci sentiamo autorizzati a diffondere indirizzi senza la previa autorizzazione delle persone interessate.
Poiché sei molto interessato a questo scambio, e la cosa ci sembra giusta e simpatica, invitiamo tutti quelli che desiderano mettersi in contatto con te, abitanti nelle provincie di Catania e Siracusa, a scrivere presso la redazione chiedendo il tuo indirizzo che, dietro tua autorizzazione, forniremo.

Se hai qualche problema e vuoi una consulenza rapida telefona in redazione ogni mercoledì pomeriggio al numero 02/797830 dalle 15 alle 18: l'esperto è a tua completa disposizione.

UN FORMATO INESISTENTE?

Ho tentato di usare il programma «DemoMaker», incluso sul dischetto 23 di AmigaByte; tutte le volte che cerco però di caricare un'immagine per incorporarla nel demo che voglio creare, il programma risponde «Sorry: picture must be 320 x 200 x 4». Ma questa risoluzione non esiste in nessun programma di grafica in mio possesso. Come posso fare?

Sandro Miracoli - Bassano del Grappa (VI)

Il formato richiesto da «DemoMaker» per le immagini IFF è 320 pixel di larghezza per 200 di altezza, con 4 bitplane; in parole povere, esso equivale al formato standard di un'immagine in bassa risoluzione NTSC con 16 colori.

Se si impiega un programma di grafica che sfrutta la maggiore risoluzione del formato video PAL europeo, le immagini salvate sono alte 256 pixel e non 200. La soluzione consiste generalmente nel salvare l'immagine desiderata sotto forma di brush, avendo cura di non superare le dimensioni massime consentite. Le brush così salvate possono essere anche più piccole del formato massimo: ad esempio, si può usare un'immagine a soli 8 colori di 310 x 180 pixel senza problemi.

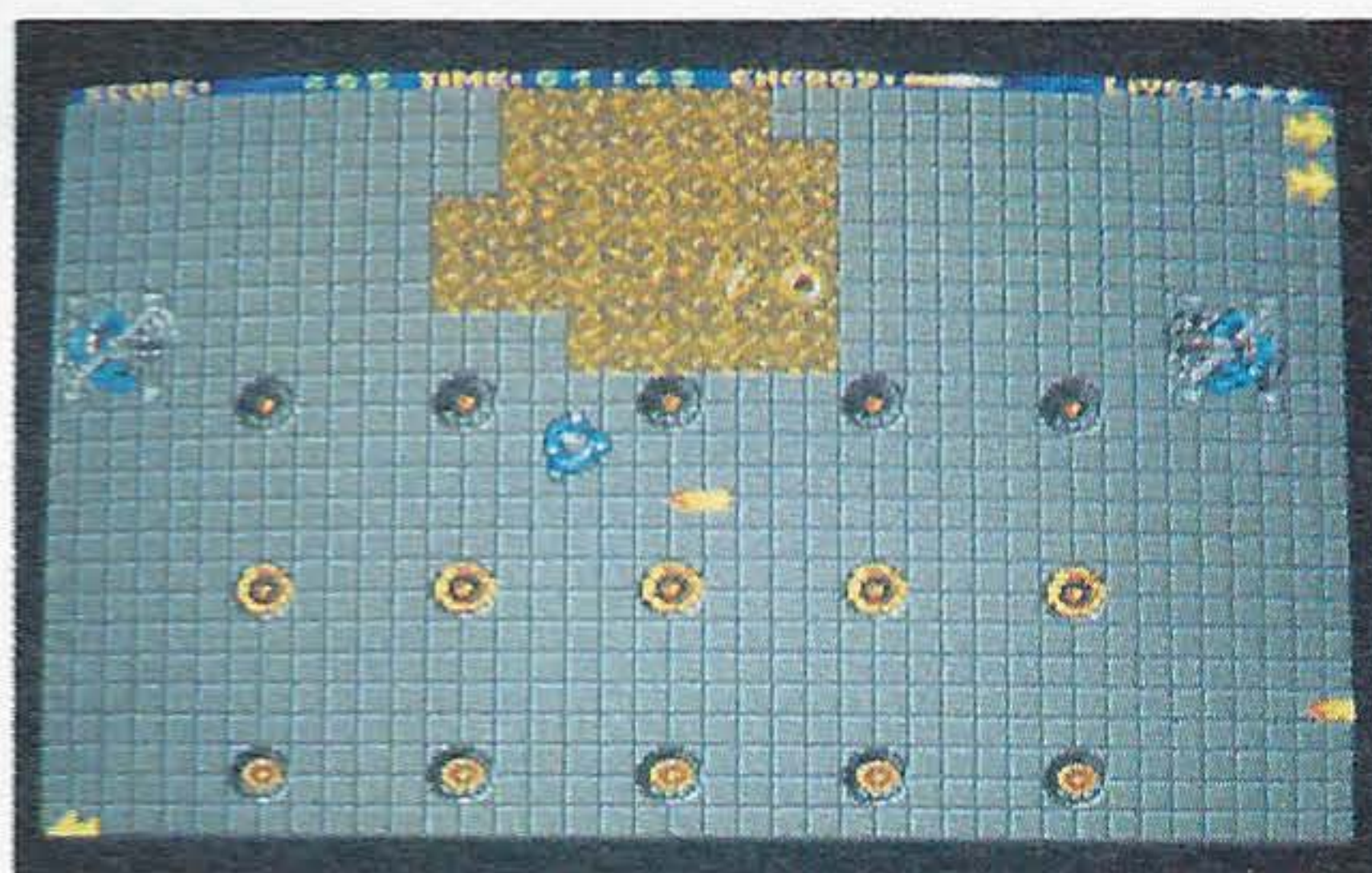
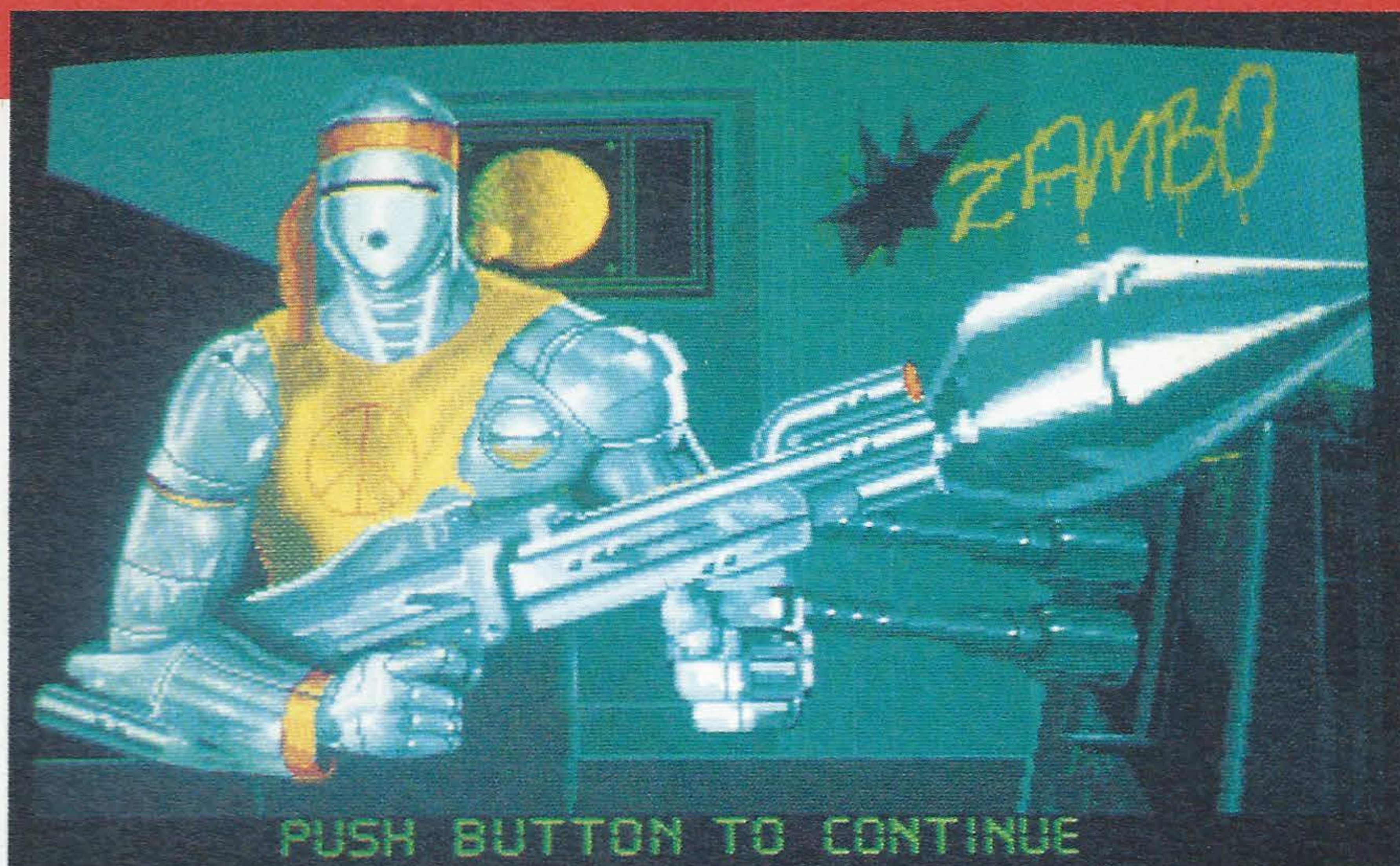
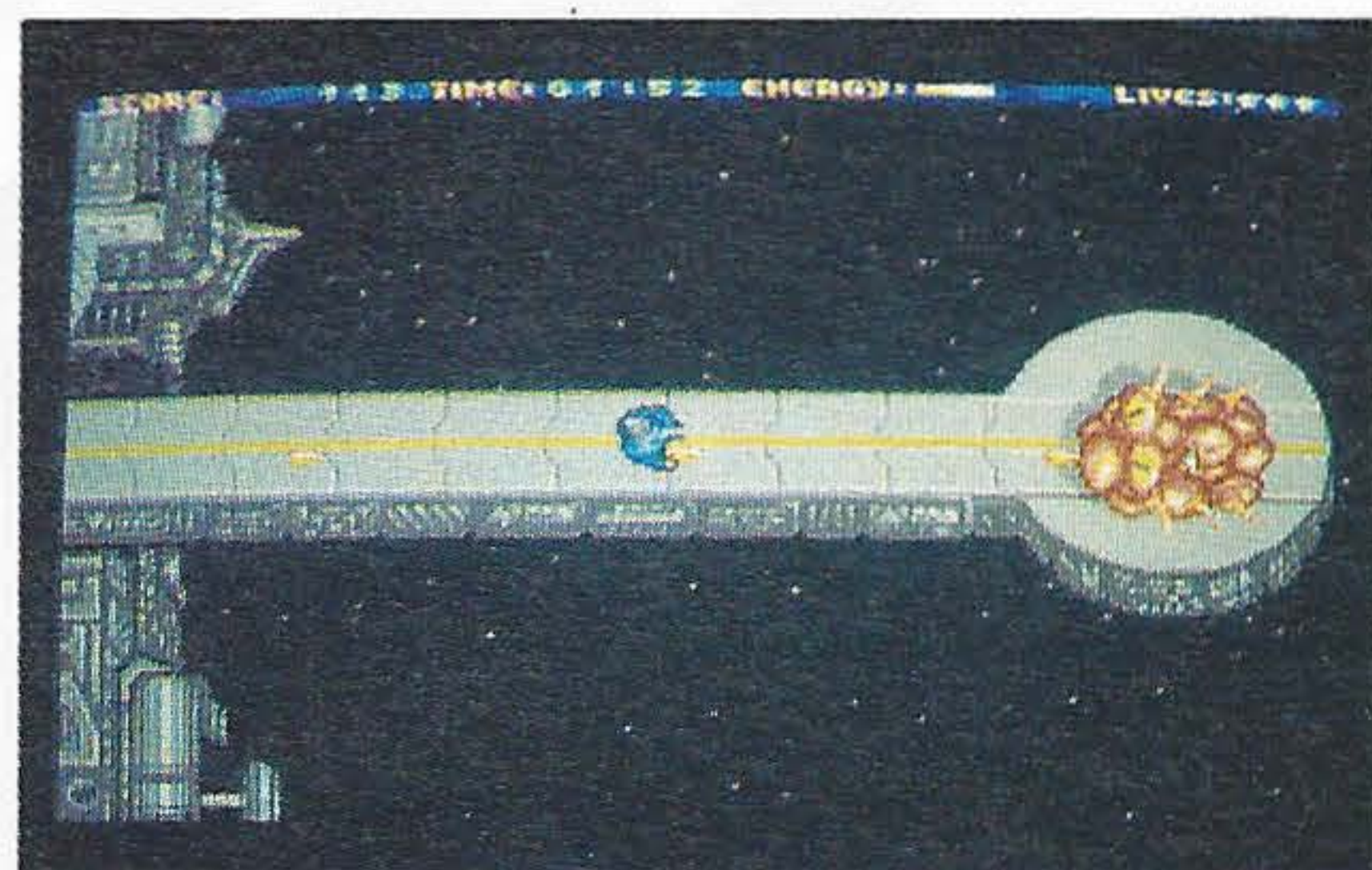
Usando «Deluxe Paint III» si può ricorrere al comando «Coords» del menu «Prefs» per tenere sotto controllo le dimensioni in pixel del brush mentre lo si sta selezionando. Sempre «Deluxe Paint III» consente inoltre, tramite l'opzione «Screen Format», di scegliere se lavorare in modalità PAL o NTSC, e di impostare il numero di colori a 16.

Come regola generale, infine, è consigliabile specificare il nome dell'immagine da caricare completo di path, ed evitare di usare spazi all'interno dei nomi. Ad esempio: «Df0: lores/immagine».

Software Express

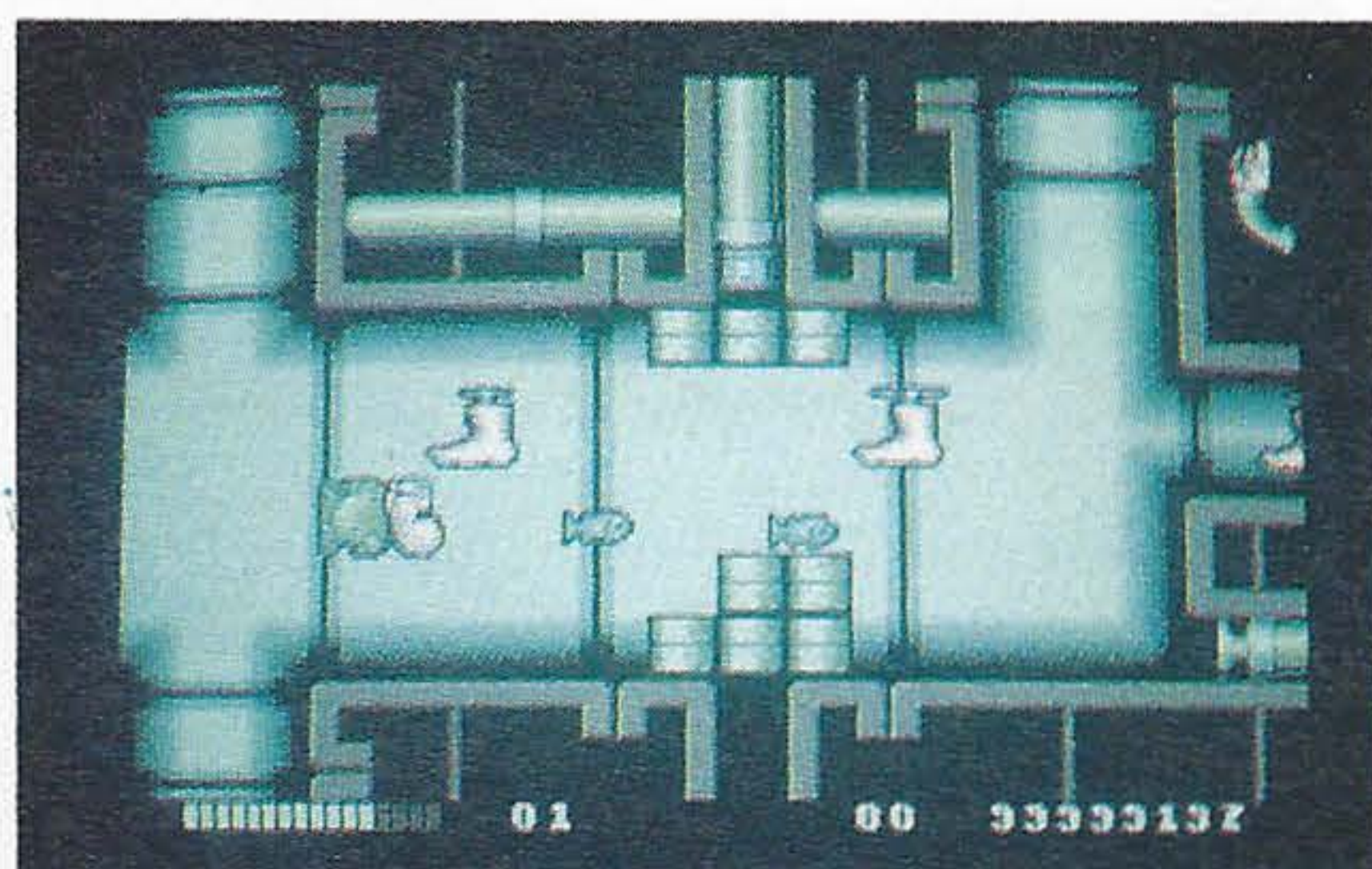
a cura di Marco Brovelli

LEAVIN' TERAMIS

Siete soli a bordo di una gigantesca astronave alla deriva nello spazio. Non proprio soli, in realtà: anche se non ci sono altri umani a farvi compagnia, la nave è popolata da un vasto assortimento di alieni e di robot decisi ad accogliervi calorosamente e a non lasciarvi più ripartire. Naturalmente, voi non siete dello stesso parere e, armati inizialmente di un misero laser, dovete percorrere tutto lo scafo evitando i colpi dei nemici o il contatto con le creature aliene e raggiungendo infine il tradizionale supermostro finale, distrutto il quale potrete passare al livello successivo. Se la trama non è certo inedita, non aspettatevi novità nemmeno dalla realizzazione: «Leavin' Teramis», tolta l'ambientazione spaziale, non è altro che un'ennesima versione di «Gauntlet», molto curata sotto il profilo della grafica e della giocabilità ma decisamente inesistente sotto quello dell'originalità.
L'azione è vista dall'alto, ed il vostro sprite occupa una porzione minuscola del piano di gioco che scrolla in tutte le direzioni. Raccogliendo i bonus, generalmente nascosti dietro a porte chiuse o in movimento, potete fare rifornimento di energia, di vite extra o di armi più potenti. Occasionalmente, seguendo le frecce sul pavimento, potete andare a prendere un po' d'aria (si fa per dire) fuori dallo scafo, raggiungendo alcune piattaforme esterne sulle quali normalmente sono annidate orde di alieni a forma di bolla.
Che altro dire? Che la grafica ed il sonoro sono più che buoni, che l'animazione e la giocabilità sono altrettanto valide, ma che il gioco fallisce miseramente nel raggiungere il suo scopo: tenere desto l'interesse del giocatore e farlo divertire. Un'altra ottima occasione sprecata per mancanza di idee.

MONTY PYTHON'S

FLYING CIRCUS

La fama del gruppo di umoristi britannici denominato «Monty Python» ha recentemente superato anche i patri confini, dove godono di un seguito entusiastico, per giungere fino a noi grazie anche a film come «Brazil», «Il senso della vita» o «Un pesce di nome Wanda», alla cui realizzazione hanno collaborato più o meno direttamente.
Il loro umorismo surreale permea anche questo videogame ad essi ispirato, per il quale anche il termine «bizzarro» è inadatto a descriverlo adeguatamente. Il muscoloso protagonista ha perso letteralmente il cervello, diviso in quattro parti, ed il suo compito è ovviamente quello di recuperarlo. Per qualche oscura ragione, l'unico modo consiste nel raccogliere 64 scatolette di carne in scatola, disseminate lungo i vari livelli del gioco.
Il mondo in cui la vicenda è ambientata è ovviamente assurdo e popolato da insidie quali pappagalli morti, gatti che esplodono, segnali stradali, poliziotti inglesi, l'inquisizione spagnola, il Ministero delle Discussioni Inutili, piedi giganteschi ed incudini che cadono. Come se non bastasse, dovete affrontare metà del gioco trasformati in un pesce (che mantiene comunque la testa in forma umana), e l'unica arma a vostra disposizione consiste nel lanciare altri pesci all'indirizzo dei nemici.

Se il tutto non vi sembra ancora abbastanza surreale, pensate che per tutta la durata del gioco l'azione viene spesso interrotta da annunci pseudo educativi, del tutto inutili, che insegnano a riconoscere a distanza diverse specie di alberi o parti del corpo.
Non si può negare a «Monty Python» la palma di gioco più originale del mese: per il resto il programma è molto curato graficamente, abbastanza giocabile e, per chi apprezza questo genere di umorismo bizzarro, decisamente divertente.

Torvak the Warrior

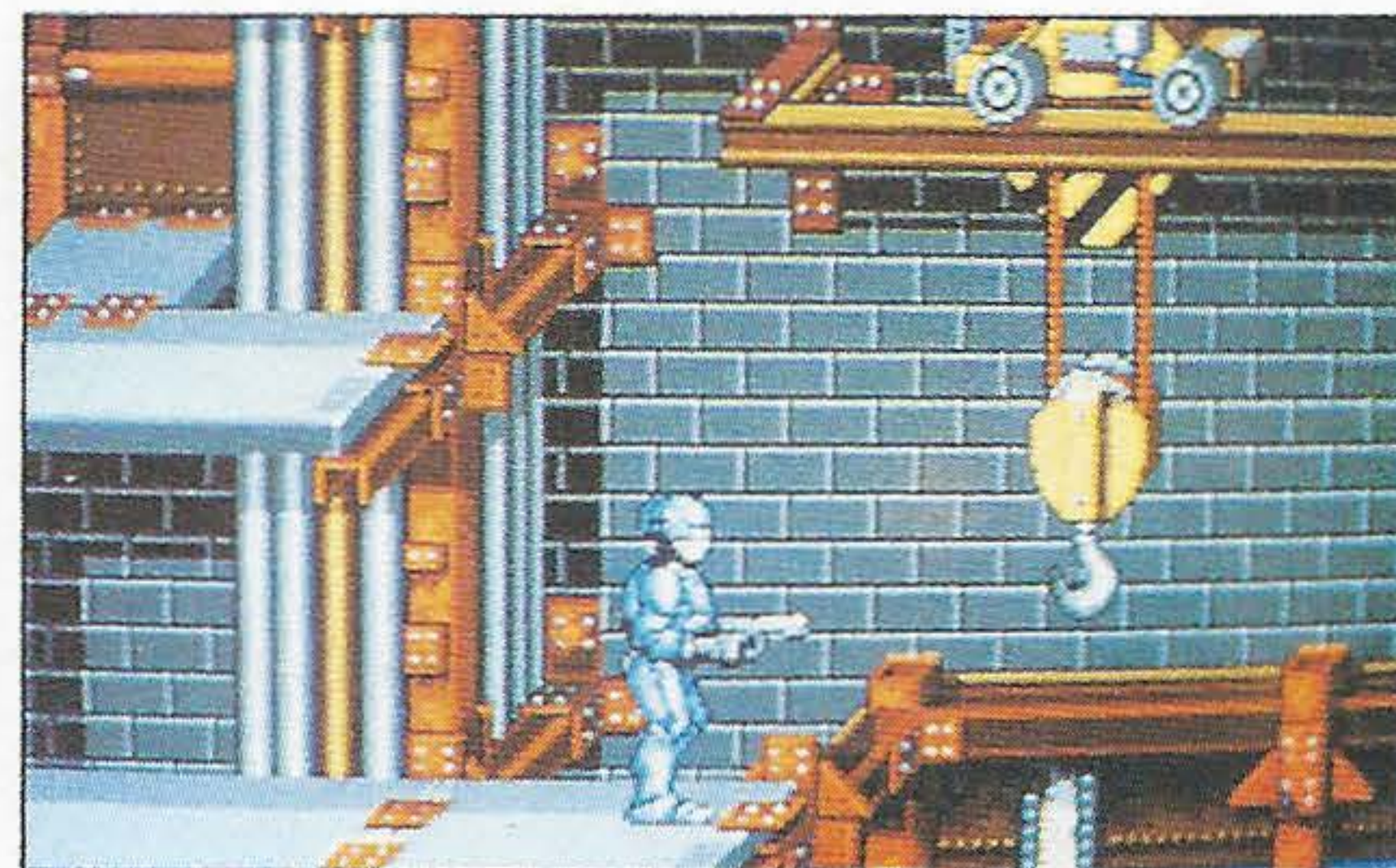
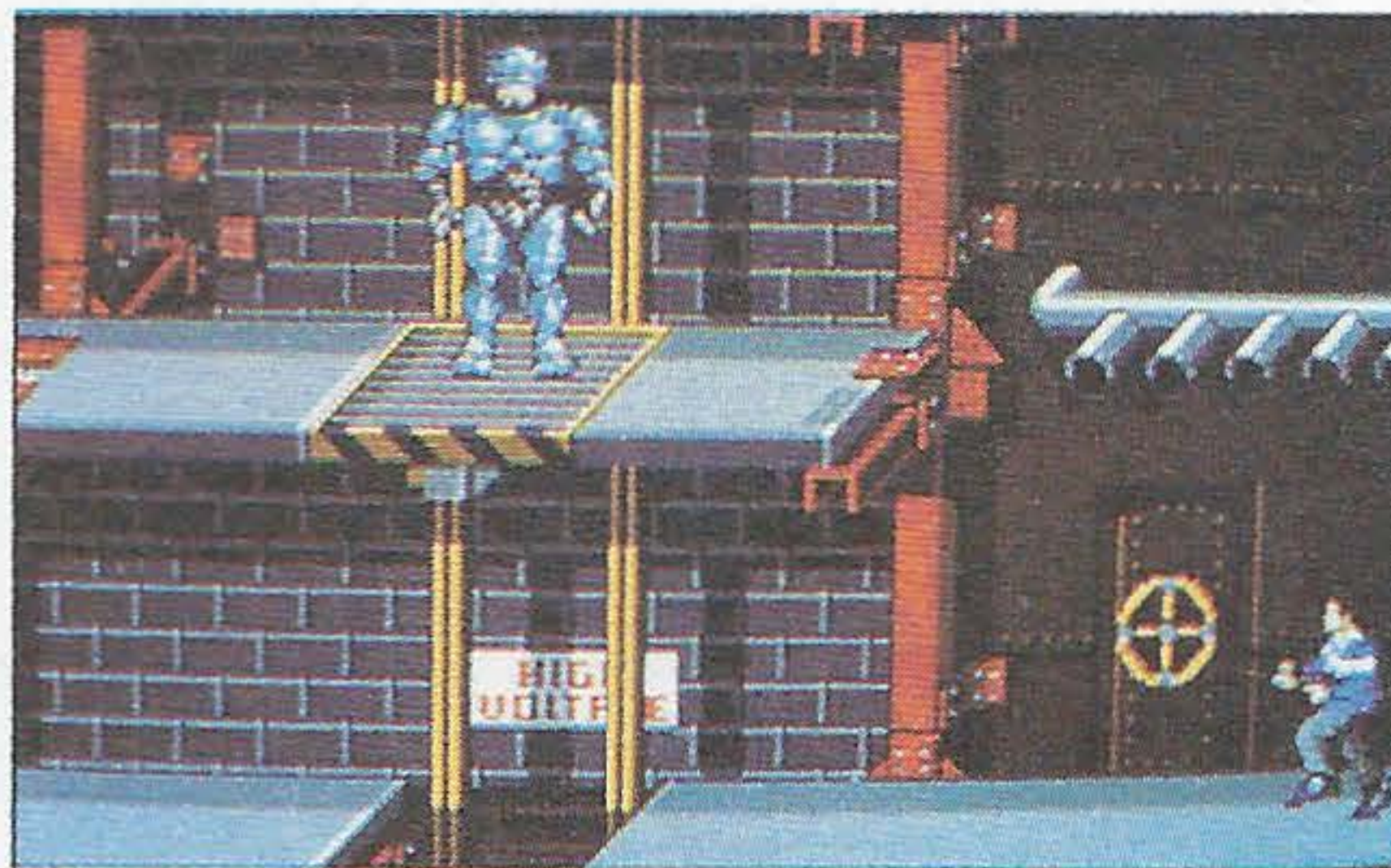
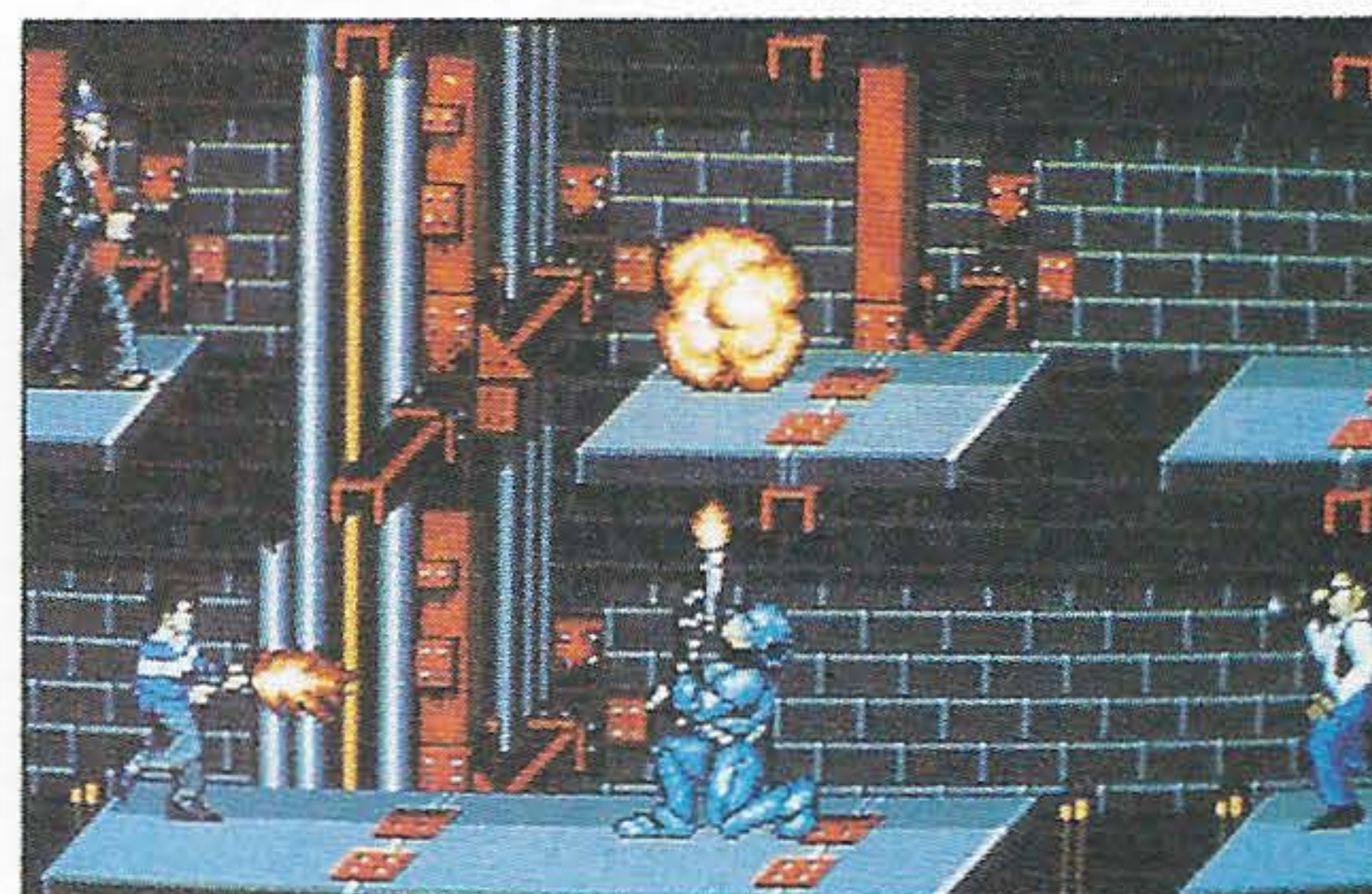
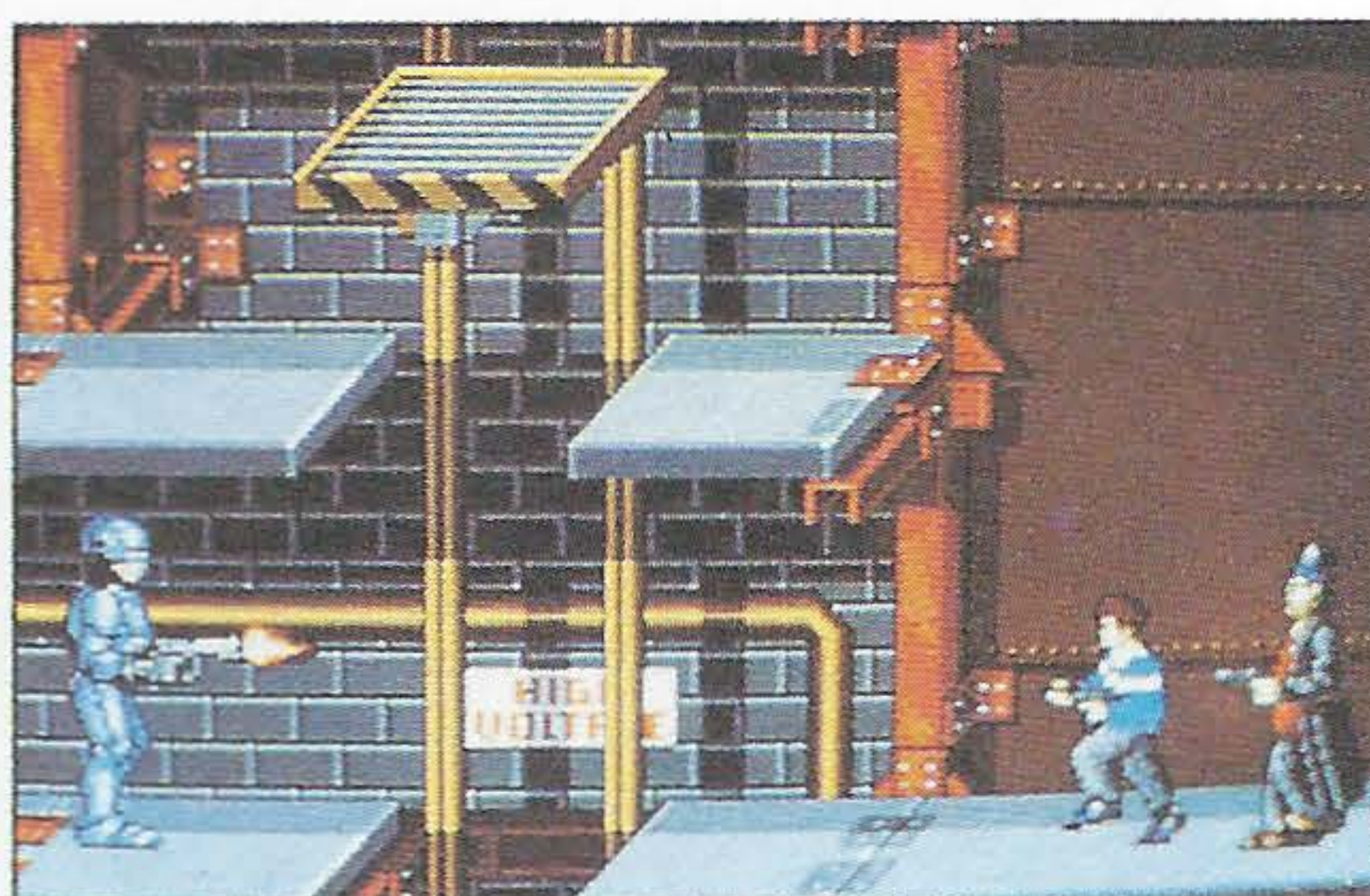
I barbari dei videogiochi sembrano tutti essere figli illegittimi di Arnold Schwarznegger: capelli neri e lunghi, sguardo truce e bicipiti che sembrano essere sul punto di esplodere. Anche Torvak, il guerriero protagonista di questo platform game della Core Design, non fa eccezione: la sua circonferenza toracica è superiore a quella di Serena Grandi, e si aggira per i 600 schermi che compongono questo discreto videogame brandendo asce e spade con la stessa disinvoltura con la quale una persona normale impugnerebbe un ombrello.

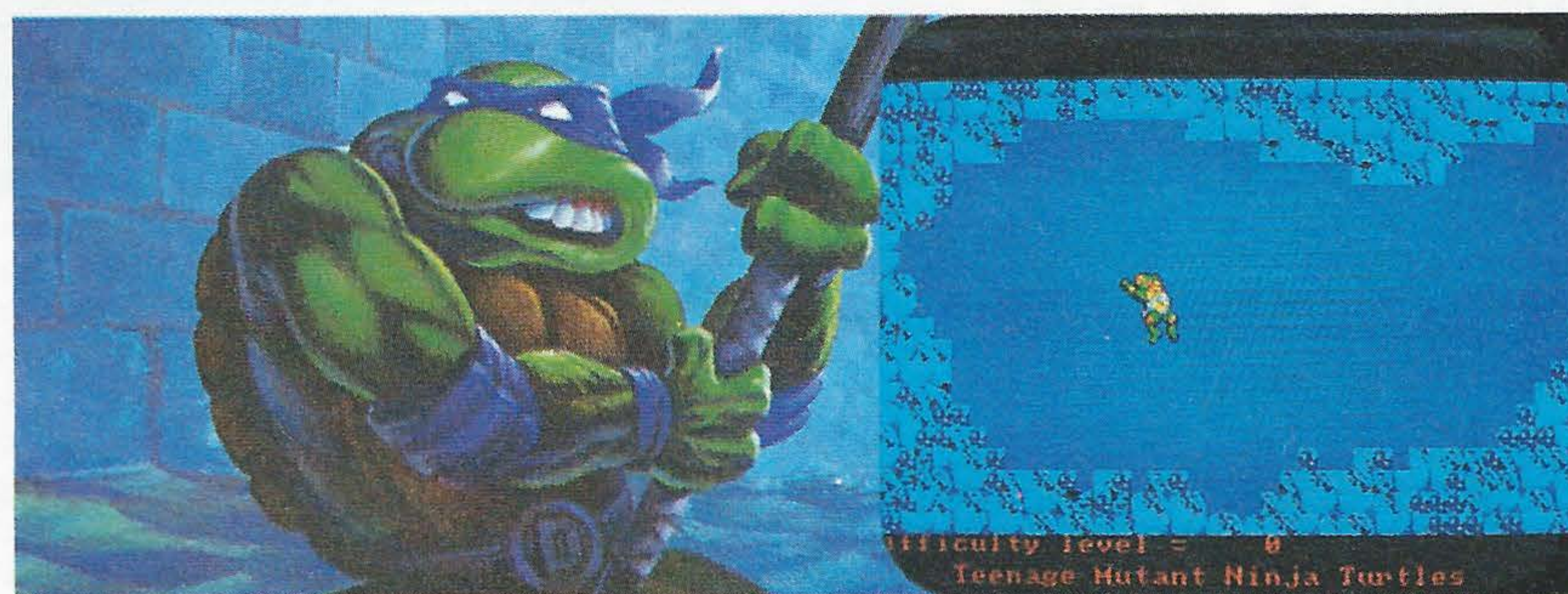
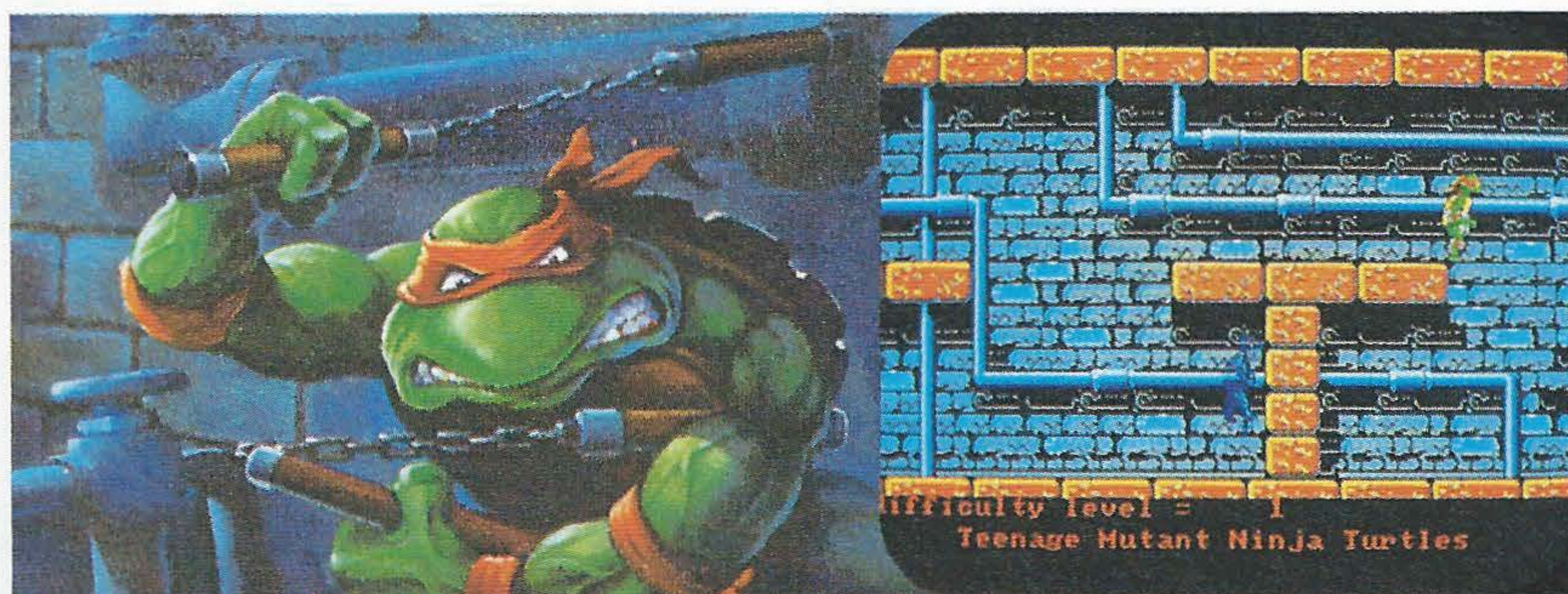
L'ormai tradizionale assortimento di mostri più o meno inferociti accompagna il barbaro nelle sue scampagnate, su e giù per pozzi, caverne e anfratti bui. Lungo il percorso sono poste diverse lapidi che, una volta fatte a pezzi a colpi d'ascia, rivelano tesori, globi di energia ed altri bonus.
Niente di nuovo sotto il sole dunque: al contrario, sotto molti aspetti, «Torvak the Warrior» rappresenta un bel passo indietro rispetto allo standard qualitativo al quale giochi simili, ma molto più curati graficamente, come «Shadow of the Beast» ci avevano abituato. L'animazione dei personaggi è ben fatta ma gli sfondi sono piuttosto monotoni e monodimensionali (un solo livello di scrolling invece dei tradizionali scenari con scrolling proporzionale che danno l'impressione della profondità).
Gli amanti del genere possono trovare in «Torvak» un passatempo divertente anche se poco originale; a chi pretende qualcosa di più (considerato anche il prezzo) consigliamo di attendere l'ormai imminente uscita del seguito del sopracitato classico della Psygnosis, intitolato «The Shadow Deepens».

U.S.S. John Young

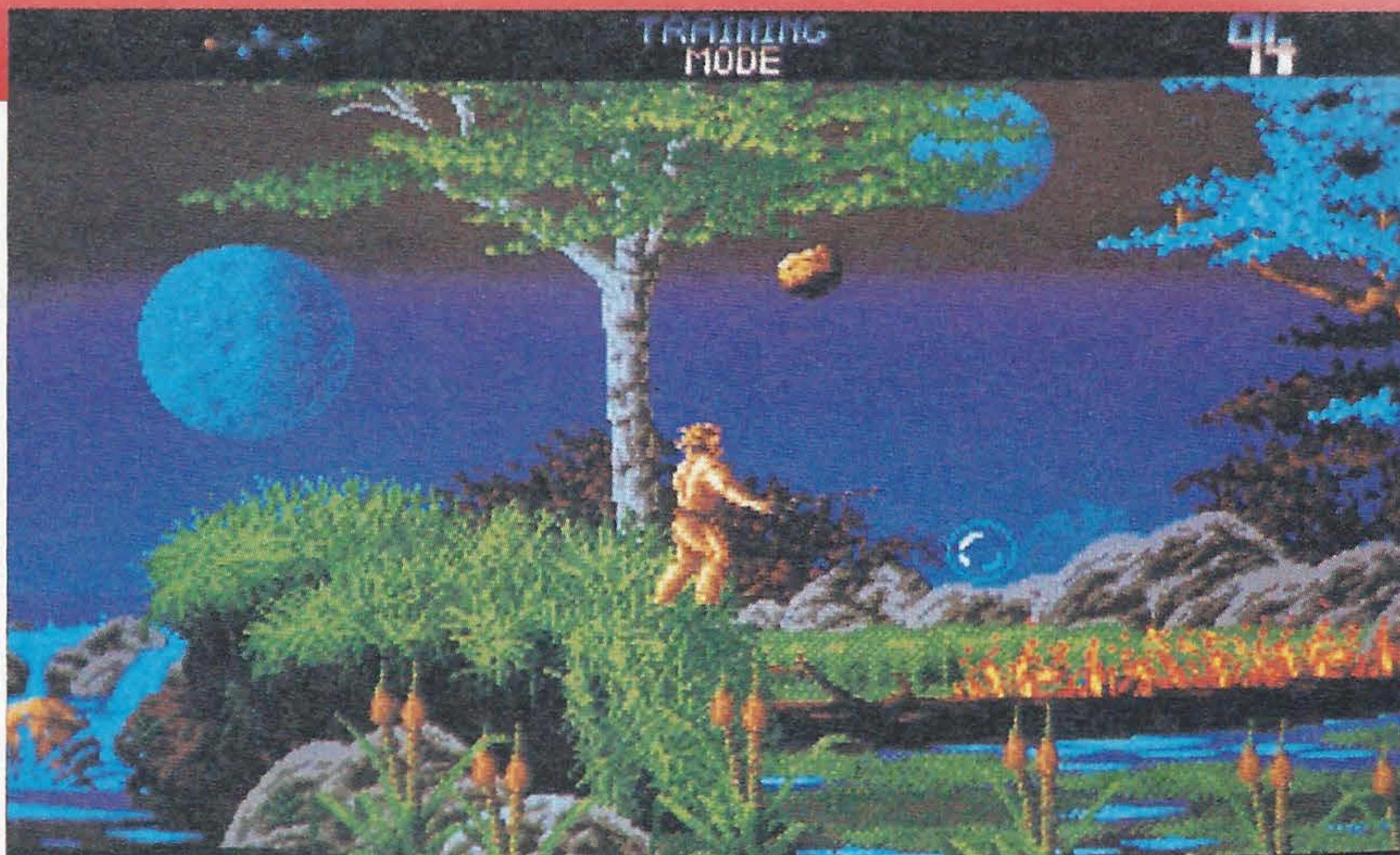
Il Golfo Persico e le regioni limitrofe, già più volte assurte agli onori della cronaca per le vicende dell'invasione irachena del Kuwait, tornano in auge come scenario di un eccellente gioco di simulazione marittima di guerra, di produzione tedesca.
La «U.S.S. John Young» del titolo è infatti un'unità navale da guerra americana, al comando della quale dovete portare a termine una serie di missioni che variano dall'affondamento di petroliere complete di scorta all'individuazione ed intercettazione con cariche di profondità di sommergibili atomici. Inutile dire che giochi come questi non fanno certo propaganda alla causa della distensione e della diplomazia ma, considerato il buon livello qualitativo di questa simulazione della Magic Bytes, è il caso di sorvolare sulle implicazioni morali e pensare invece a divertirsi, navigando al largo dell'Iran lungo lo stretto di Hormutz e lanciando cannonate a tutto spiano.
La grafica, anche se non eccezionale, è di buon livello e il meccanismo di gioco non è eccessivamente complesso, risultando assai simile a quello del classico «Silent Service».

A parte qualche appunto sull'irritante lentezza di caricamento da floppy (inconveniente che affligge il giocatore per tutta la durata della partita), non ci sono particolari critiche da muovere nei confronti di «U.S.S. John Young». Ai più esigenti il livello di difficoltà potrà apparire fin troppo elementare, ma in generale la giocabilità e la varietà di schermi ed opzioni non dovrebbero mancare di soddisfare coloro che vogliono solo trascorrere piacevolmente un paio di ore indossando i panni dell'ammiraglio.

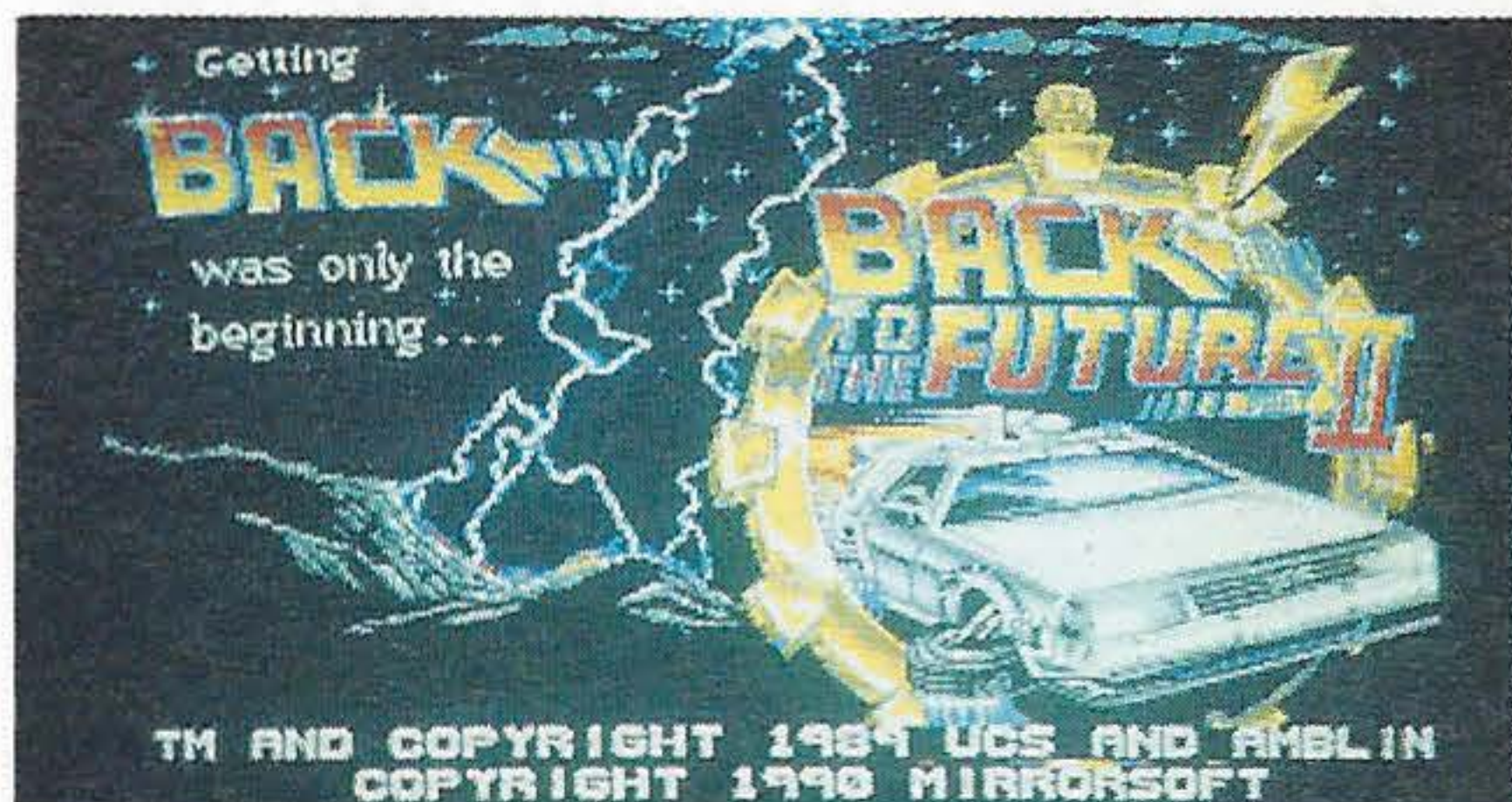
Il videogame tratto dal primo film «Robocop» ha battuto tutti i record di popolarità, risultando al primo posto nella classifica del software a 16 bit più venduto della storia. Naturale che, in concomitanza con l'uscita sugli schermi europei del secondo episodio della serie cinematografica, la Ocean abbia deciso di riprovarci, proponendo un nuovo gioco arcade ispirato alle sequenze più emozionanti del film.
La meccanica e la giocabilità non sembrano essere mutate rispetto al predecessore: sullo schermo si alternano una serie di livelli a scrolling orizzontale in cui Robocop deve eliminare a colpi di pistola i vari delinquenti che sbucano da ogni direzione. Ogni livello è intervallato da sequenze-bonus ispirate ad altrettante scene chiavi del film, in cui Robocop deve risolvere entro un tempo limite particolari missioni come recuperare la memoria, ricostruendo un'immagine separata in vari frammenti sparsi all'interno del suo cervello elettronico, oppure allenarsi al tiro a segno, per migliorare la propria accuratezza nella scelta dei bersagli evitando di colpire chi non è armato.
Ogni livello ha un'ambientazione diversa (lungo le strade, in una fabbrica abbandonata, all'interno di un grattacielo, ecc.) e richiede armi e tattiche particolari per essere portato a termine. Oltre allo sterminio di tutti i deliquenti, Robocop deve riuscire in ogni livello ad arrestare alcuni personaggi chiave, raggiungendoli ed afferrandoli senza fare ricorso alle armi.
Graficamente «Robocop 2» è persino migliore del suo predecessore, del quale senza dubbio eguaglierà lo straordinario successo.

ROBOCOP 2

TEENAGE MUTANT NINJA TURTLES

Apparse negli Stati Uniti prima sulle pagine di una serie a fumetti, ed in seguito sugli schermi televisivi sotto forma di disegni animati, le «giovani tartarughe ninja mutanti» sono ora protagoniste di un platform game prodotto dalla Ultra Software. Se la popolarità di questi bizzarri personaggi nel loro Paese d'origine è piuttosto alta, non si può dire altrettanto qui da noi. Questo programma non aiuterà comunque ad aumentarne la notorietà, poiché sia la parte grafica che la giocabilità non sono nulla di eccezionale.
Le tartarughe mutanti sono quattro buffi individui dai nomi rinascimentali (Michelangelo, Donatello, Raffaello e Leonardo!), esperti nell'uso delle arti marziali. Per salvare una fanciulla rapita dal perfido Shredder devono percorrere le fognature di una città, eliminando i mostri che le infestano e saltando su e giù per tubature, piattaforme e scale. Ogni tartaruga è esperta nell'uso di una particolare arma (spada, shuriken, nunchaku o bastone), ed è possibile alterarne il controllo di ogni personaggio per sfruttarlo adeguatamente a seconda delle situazioni.
I protagonisti del gioco sono molto buffi, e probabilmente chi già li conosceva attraverso i fumetti o la televisione si divertirà a controllarne le azioni sullo schermo. Dal punto di vista qualitativo, «Teenage Turtles», non raggiunge però neanche la sufficienza: la grafica è appena discreta, e l'azione troppo limitata per tenere desto a lungo l'interesse del giocatore.

Software Express

UNREAL

Provate ad immaginare un incrocio tra l'ambientazione e la grafica di «Shadow of the Beast» con la meccanica delle sequenze di volo in prospettiva di «Space Harrier» ed avrete un'idea abbastanza aderente alla realtà dell'aspetto di «Unreal», l'ultimo, e fino ad ora migliore, prodotto della francese Ubisoft.
Veniamo subito al dunque: «Unreal» graficamente è tra i migliori videogiochi mai visti su Amiga. La fluidità delle animazioni ed il numero di colori e di sprite in movimento sullo schermo lasciano talvolta a bocca aperta il giocatore. La trama è, come di consueto, banale e del tutto inutile: la vostra ragazza è stata rapita dal solito demone malvagio e dovete andare in suo soccorso percorrendo svariati livelli a scorrimento orizzontale, risolvendo gli eventuali problemi incontrati lungo la strada e facendo a fettine tutte le creature che cercano di sbarrarvi la via.
Tra un livello e l'altro, vi spostate invece a cavallo di un drago volante, manovrandolo in modo da evitare di sbattere contro gli ostacoli che vi sfrecciano contro e raccogliendo eventualmente i bonus per aumentare l'energia. Sono queste sezioni tridimensionali a lasciare particolarmente stupefatti per la velocità dei movimenti e la bellezza degli sfondi; unico neo è la loro durata, a volte decisamete eccessiva e sproporzionata rispetto a quella degli altri livelli.
Musica ed effetti sonori sono altrettanto superbi: «Unreal» sfrutta così a fondo le potenzialità dell'hardware di Amiga da non sembrare vero (come dice il nome stesso). Una perfetta dimostrazione di ciò che si può ottenere da Amiga quando il gioco è pensato in funzione delle capacità del computer invece di essere una stanca conversione di software sviluppata su qualche altra macchina inferiore.

NEUROMANCER

Distribuito dalla Electronic Arts e prodotto dalla InterPlay, software house specializzata in giochi di avventura e di ruolo (tra i quali ricordiamo il celeberrimo «Bard's Tale»), arriva finalmente anche sugli schermi di Amiga la versione computerizzata del romanzo di fantascienza di William Gibson «Neuromancer», ritenuto l'iniziatore del genere narrativo chiamato «Cyberpunk».
I computer e gli «hacker» rivestono un ruolo predominante in questa vicenda ambientata in un futuro in cui hardware e software interagiscono direttamente con il cervello dei loro utilizzatori, calandoli in un mondo artificiale simulato definito «cyberspace».
In questa avventura arcade (totalmente controllata tramite il mouse) il giocatore interpreta il ruolo di Case, un futuristico hacker particolarmente abile nell'aggirare le protezioni e i dispositivi di sicurezza delle più complesse reti di computer.
All'inizio del gioco vi svegliate in bar, senza denaro o informazioni dettagliate su quale sia la vostra missione. Non è difficile tuttavia fare progressi ed esplorare gran parte del mondo di «Neuromancer» alla ricerca di indizi.
Ad ogni mossa falsa verrete imprigonati, processati e multati (o condannati a morte): niente paura, comunque, poiché in caso di necessità potete sempre guadagnare qualche soldo vendendo i vostri organi vitali alla clinica dei trapianti...
Grafica e sonoro non brillano particolarmente in «Neuromancer», ma la trama è molto avvincente!

BACK TO THE FUTURE II

Nei cinema è ormai praticamente in uscita la terza parte della saga cinematografica di «Ritorno al futuro», ma i programmatori della MirrorSoft sono evidentemente un po' indietro con la tabella di marcia e propongono solo ora la versione computerizzata del secondo film della serie. Non ci sarebbe nulla da obbiettare se la qualità del prodotto fosse proporzionale al tempo necessario per farlo uscire, ma sfortunatamente così non è: «Back to the Future II» è un gioco simpatico, a tratti divertente ma che decisamente non sfrutta il potenziale offerto dalla geniale trama del film di Robert Zemeckis.

I cinque livelli del gioco sono ispirati ad altrettante sequenze chiave del film; le più interessanti risultano essere la prima e l'ultima (del tutto identiche a parte l'epoca di ambientazione), nelle quali il protagonista Marty McFly deve sfuggire ad un'orda di inseguitori a bordo di uno skateboard, evitando ostacoli ed automobili e raccogliendo i bonus sparsi lungo la strada, in uno stile molto simile a quello del gioco «Paperboy».

I rimanenti livelli sono più macchinosi e meno divertenti, e vedono Marty alle prese con la fidanzata Jennifer in un ospedale o mentre fa a cazzotti con il cattivo Biff; il peggiore in assoluto è il quarto, una specie di rompicapo ispirato al «gioco del 15», in cui dovete riordinare correttamente le tessere che compongono un'immagine di Marty mentre suona la chitarra prima che scada il tempo.

La cosa migliore di «Back to the Future II» è senza dubbio la sequenza animata iniziale, direttamente tratta dal film, che mostra la macchina del tempo mentre sfreccia verso il futuro. Ma il gioco, nonostante la discreta grafica, non è sufficientemente giocabile ed avvincente, e si rivela una cocente delusione per tutti gli ammiratori del film.

HEAVY METAL

A dispetto del titolo, questo programma della Access non ha nulla a che fare con la musica rock: il metallo al quale si fa riferimento è quello dei tre mezzi corazzati protagonisti di altrettante sezioni del gioco.

Nella prima vi trovate nella cabina di tiro di un tank e, armati di radar e cannone, dovete eliminare i carri armati nemici che circolano nelle vicende prima che essi possano fare altrettanto con voi. In questa sezione, chiaramente ispirata al classico «BattleZone», la difficoltà è rappresentata soprattutto dal dover continuamente spostare l'angolo di inclinazione e l'orientamento del cannone per colpire i mezzi nemici, il tutto il più rapidamente possibile.

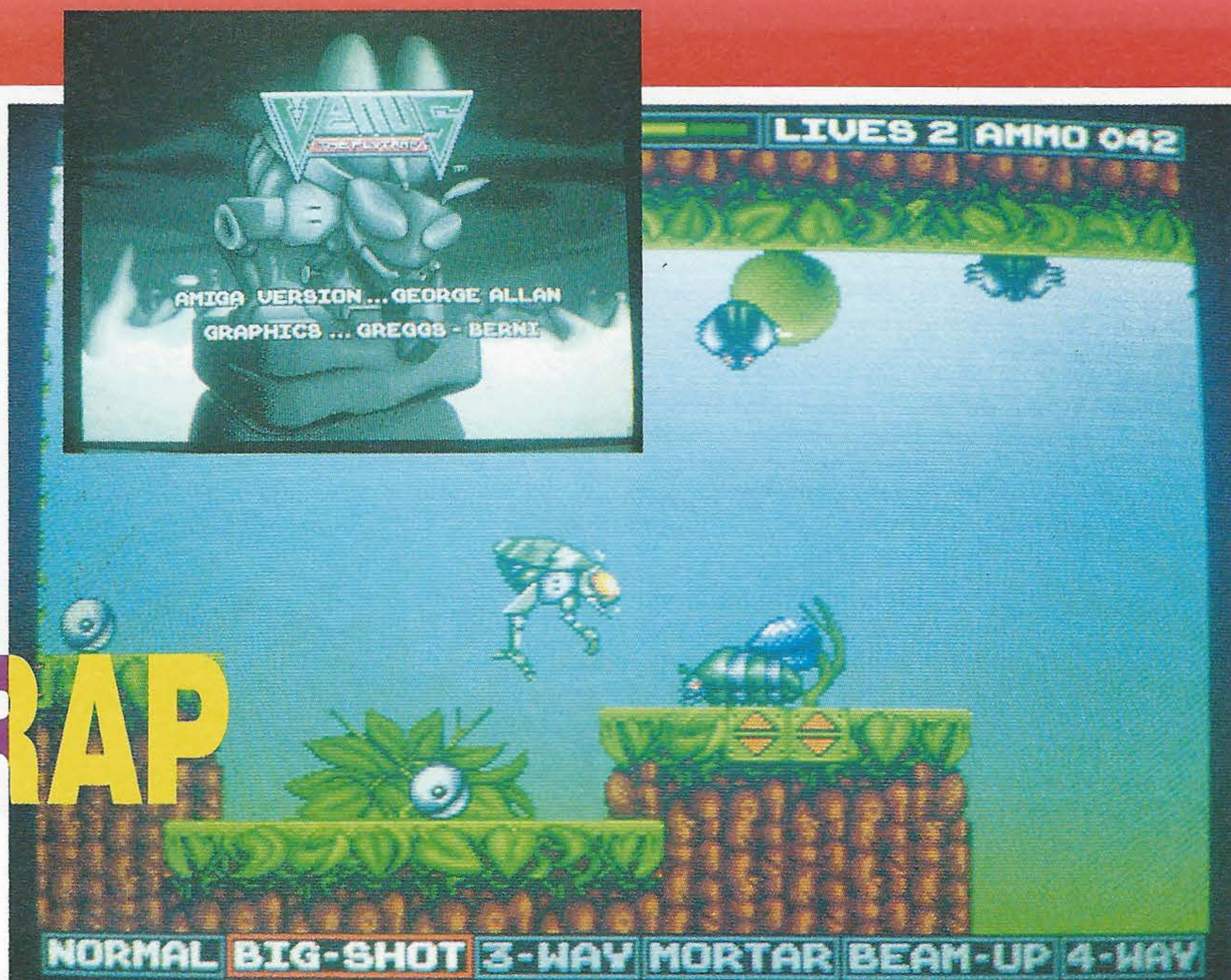
Nel secondo livello siete a bordo di un blindato in movimento lungo il deserto, e dovete fare piazza pulita delle installazioni e dei velivoli nemici a colpi di mitra. La grafica, in prospettiva, è animata ed il vostro mezzo è visto da dietro mentre sfreccia in avanti: la difficoltà è considerevolmente aumentata dall'impossibilità di rallentare o frenare.

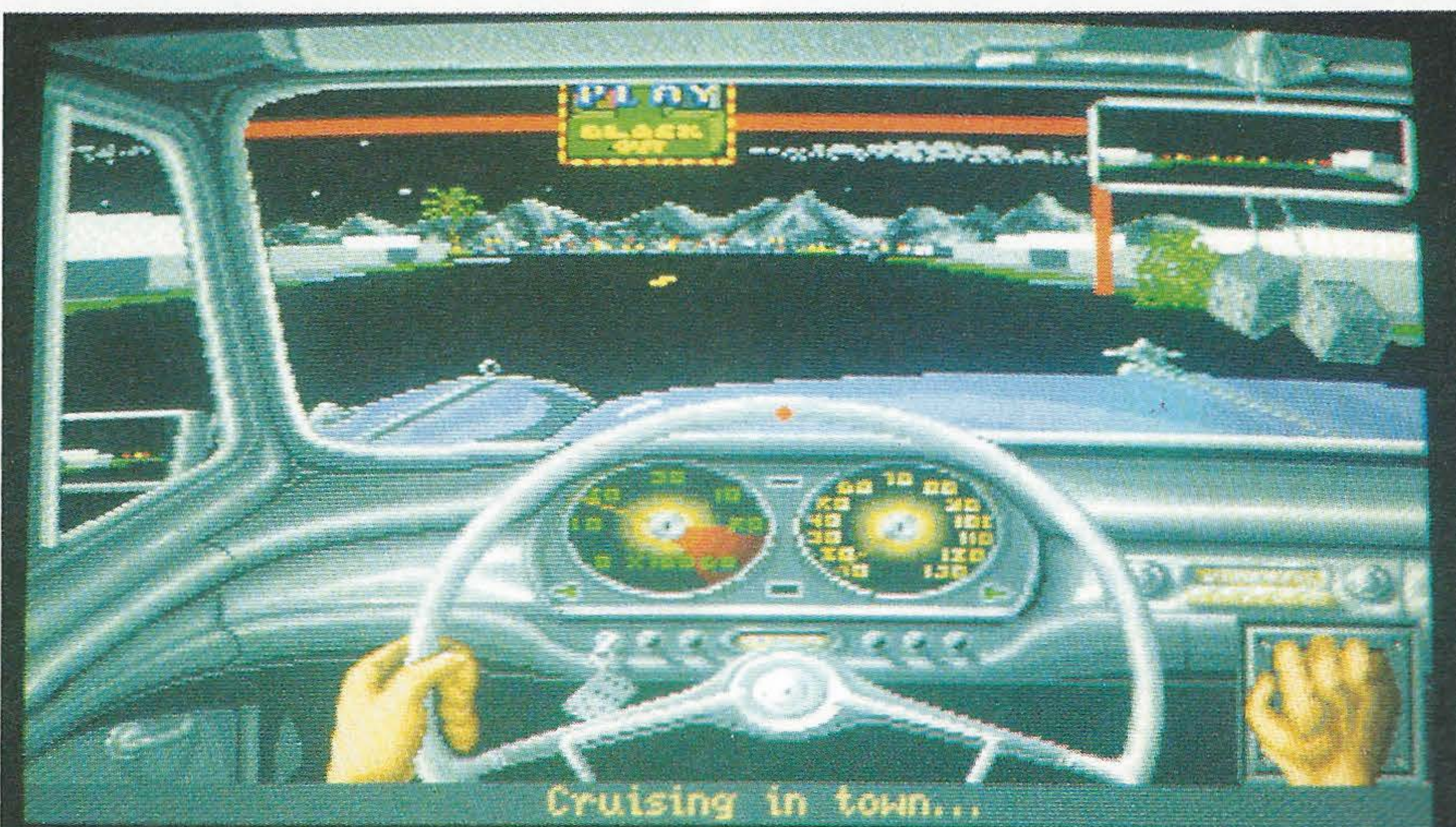
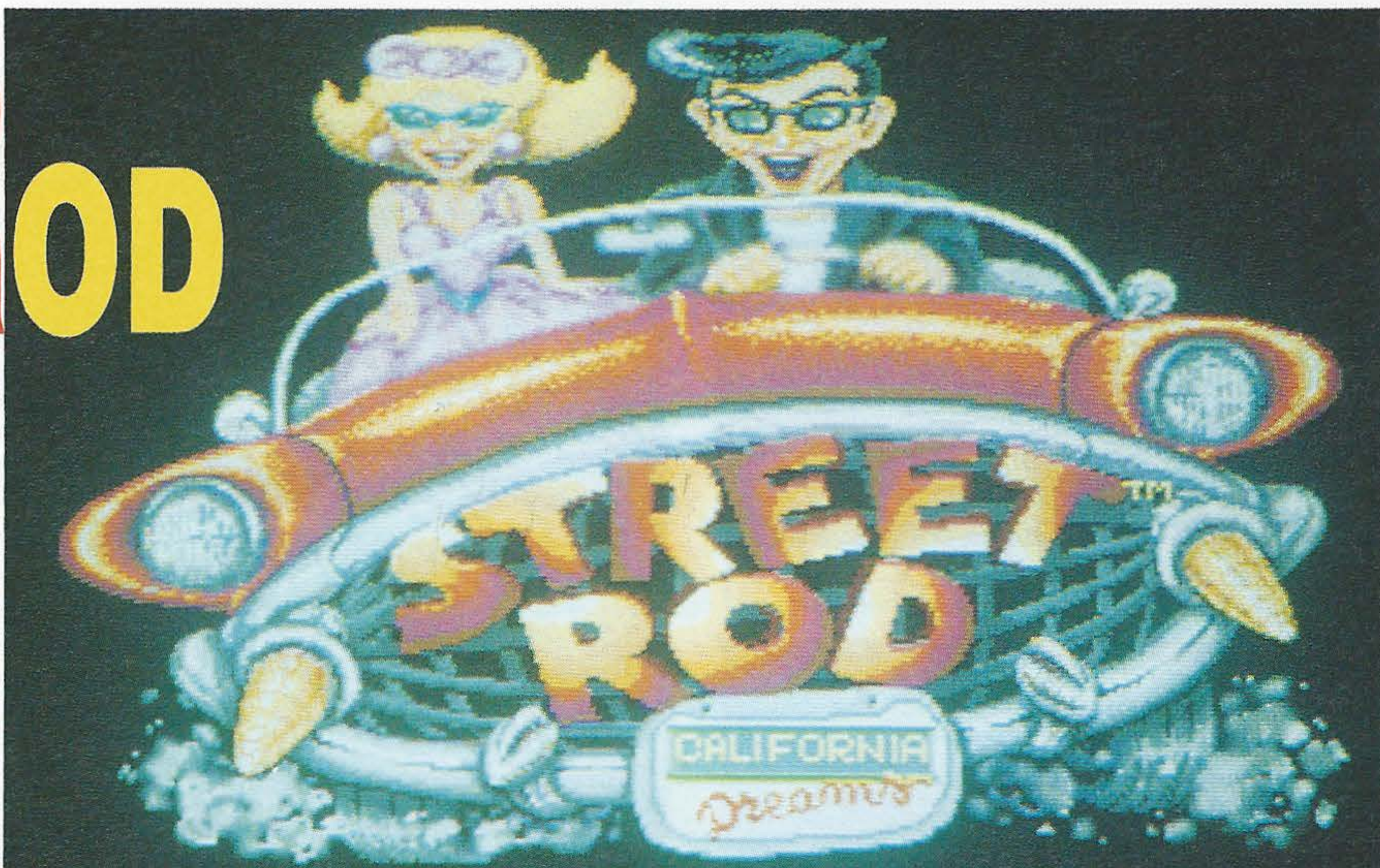
L'ultima sezione è nuovamente statica: vi trovate a bordo di un veicolo lanciamissili terra-aria e dovete abbattere il maggior numero di aerei possibile prima che il loro equipaggio vi riduca a groviera a suon di mitragliatore.

Troppo superficiale come simulazione e non abbastanza avvincente o giocabile come arcade, «Heavy Metal» manca il suo bersaglio principale: divertire il giocatore.

La grafica non è malvagia, ma l'azione diventa snervante e monotona dopo un paio di partite, soprattutto per la mancanza di variazioni tra i livelli di gioco. Se amate i tank ed i giochi di guerra, lasciate perdere «Heavy Metal» e procuratevi una copia di «Sherman M4».

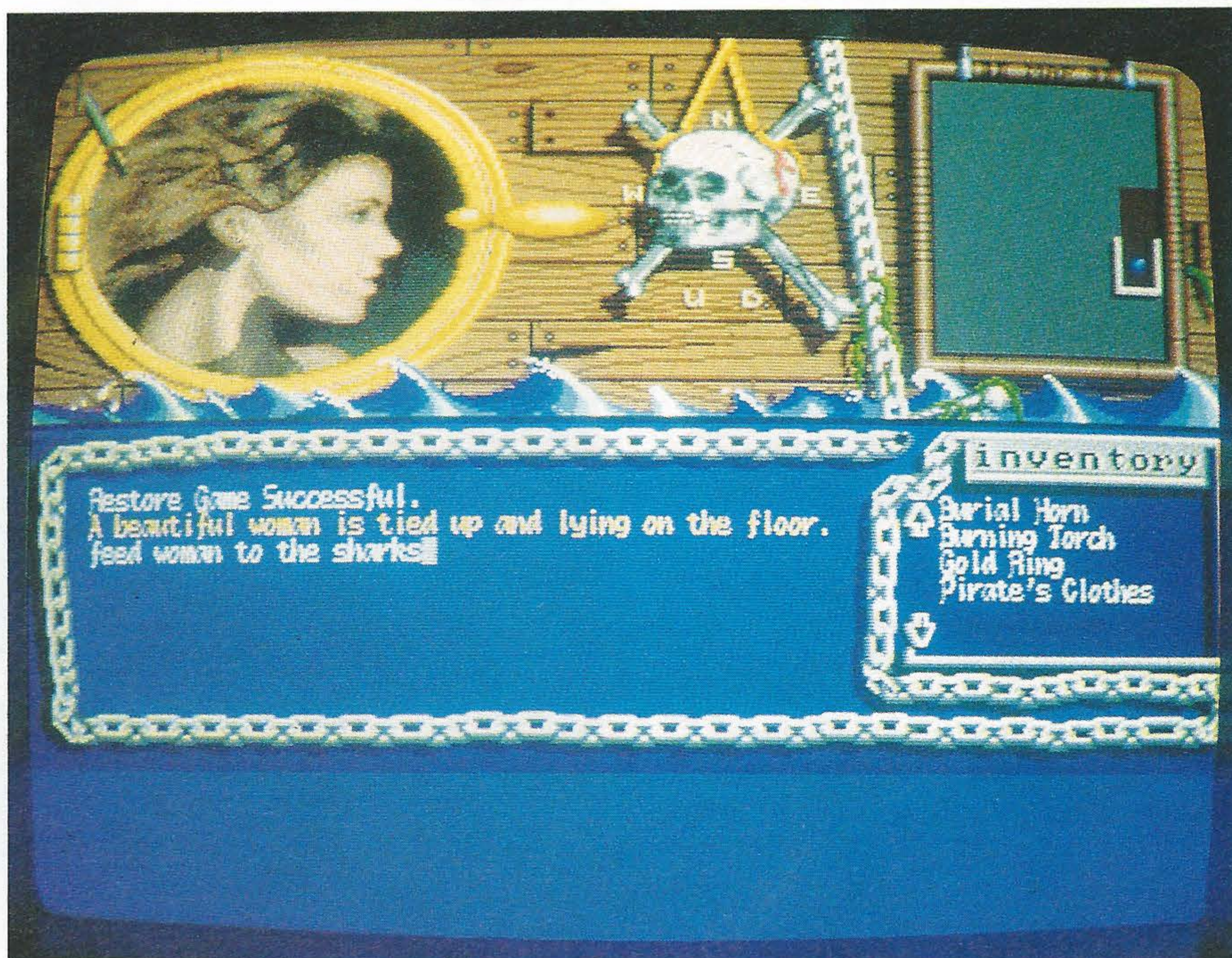
VENUS THE FLYTRAP

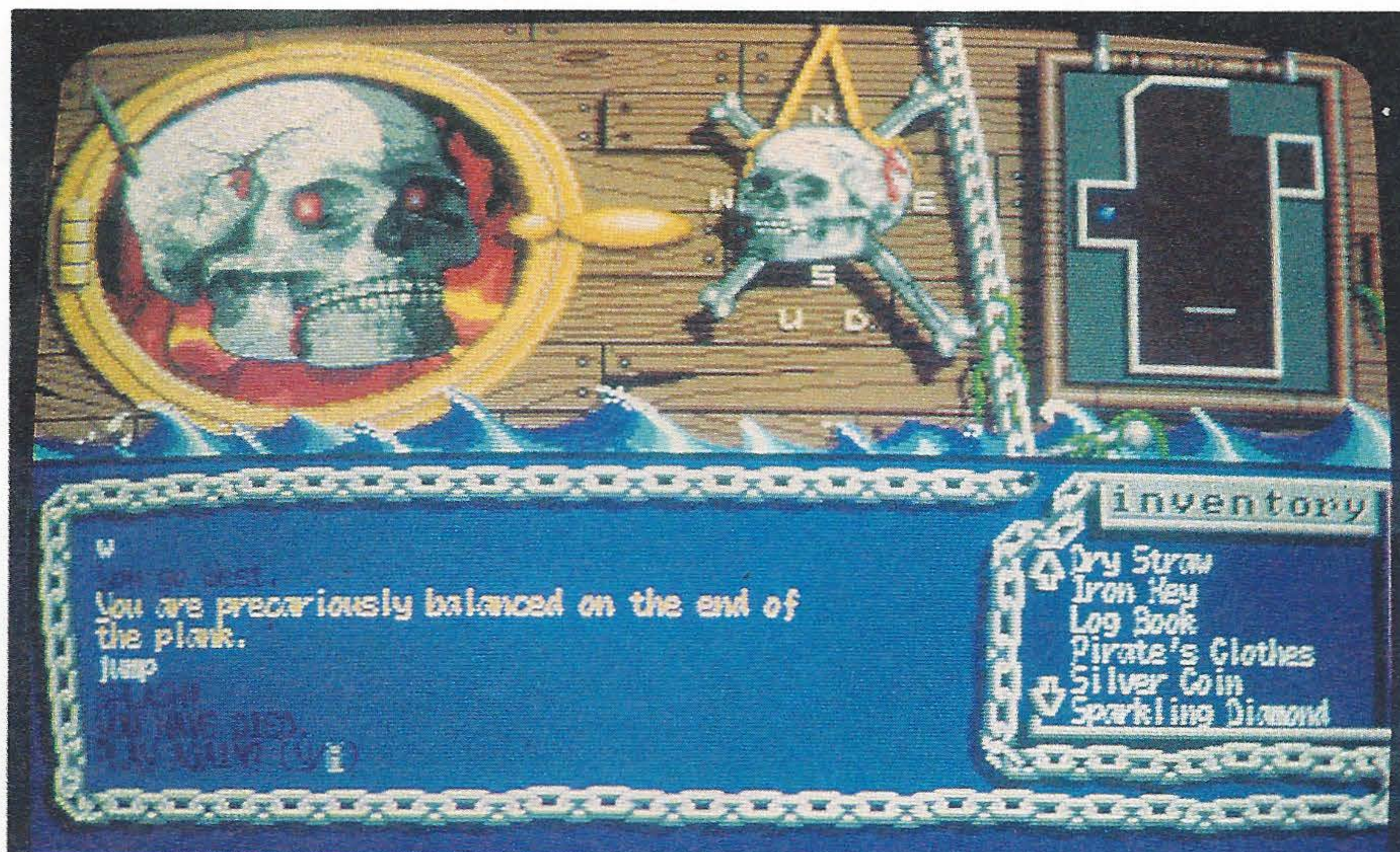
La Gremlin Graphics è una software house il cui nome è garanzia di ottimo livello qualitativo: raramente i giochi prodotti da questa casa risultano particolarmente originali o innovativi, ma alla carenza di idee nuove sopperiscono l'eleganza della realizzazione e la grande giocabilità.
Questo «Venus» conferma la veridicità di questa teoria, essendo del tutto scontato sotto il profilo della trama (un ennesimo shoot'em up a scrolling orizzontale) pur risultando giocabilissimo ed entusiasmante.
Protagonista del gioco è una mosca robotica che deve attraversare dieci mondi differenti, ognuno dei quali è composto da cinque livelli. Come di consueto, tutto ciò che si muove è un nemico e deve essere distrutto: raccogliendo i globi lasciati dai cadaveri, si possono aggiungere armi più potenti a quelle inizialmente in dotazione, oppure aumentare il punteggio, l'energia o il tempo a disposizione del nostro insetto volante (o saltellante, come è più corretto dire in questo caso).
Lo scrolling degli schermi non è continuo, ma è regolato dai movimenti del giocatore: se la mosca è immobile, anche lo schermo resta fermo per consentire al giocatore di studiare la situazione e pianificare il da farsi.
Gli sfondi sono suggestivi, essenziali e coloratissimi (sembrano quasi in modo HAM): gli sprite sono ben disegnati e animati benissimo: in particolare i movimenti della mosca, specialmente i salti e il caratteristico «grattare» il terreno con le zampe, sono molto realistici.
Un prodotto di ottimo livello, vivamente consigliato agli estimatori del genere.

STREET ROD

Avete presente American Graffiti? La provincia americana degli anni '60, con i drive-in e le Chevrolet truccate, funge da ambientazione per questo simpatico gioco della California Dreams, un misto di simulazione strategica ed azione arcade automobilistica.
All'inizio del gioco disponete di soli 750 dollari e di un vecchio giornale, attraverso i cui annunci economici potete acquistare una vecchia automobile e le parti di ricambio con cui ricarrozzarla e truccarne il motore, per trasformarla in un bolide a quattro ruote.
Per guadagnare il denaro necessario all'acquisto degli accessori, protete scendere in città, parcheggiare davanti al fast-food e sfidare qualcuno dei bulli locali ad una gara di velocità, mettendo in palio una posta in denaro o qualche pezzo di ricambio.
Oltre a dover migliorare costantemente le prestazioni della propria vettura, per riuscire a lasciare nella polvere anche i campioni più spavaldi, occorre prestare attenzione alla polizia, tradizionalmente poco tenera con chi viola i limiti di velocità, e gli altri occasionali imprevisti (guasti meccanici, sabotaggi, ecc.) che vi renderanno la vita molto difficile. Il gioco termina una volta esaurito il denaro a vostra disposizione.
La grafica è di ottimo livello, sia per quanto concerne la parte più statica del gioco (ovvero le sezioni in cui provvedete alla messa a punto della vettura) che durante le sequenze arcade in cui gareggiate.

ISLAND OF LOST HOPE

Da naufrago su un'isola deserta a capitano di una nave pirata carica di oro e gioielli, il passo è più breve di quanto sembri: se non ci credete, provate a giocare a «Island of Lost Hope» e ve ne renderete conto da soli.
Certo, le insidie da superare non sono poche: pirati inferociti, serpenti velenosi, topi giganteschi, trappole mortali...
Con un piccolo aiuto da parte nostra, potrete comunque facilmente superare tutti questi ostacoli: dopo averne infatti parlato favorevolmente sul fascicolo 22 di AmigaByte, presentiamo ora la soluzione completa di questo popolare adventure-game grafico della Digital Concepts. Come di consueto, la soluzione elenca soltanto i comandi indispensabili per completare il gioco e deve essere seguita alla lettera.

LA SOLUZIONE

CLIMB TREE - SIGNAL SHIP - SHAKE TREE - DOWN - REMOVE CLOTHES - UNDRESS PIRATE - WEAR PIRATES CLOTHES - ENTER BOAT - NORTH - WEST - NORTH - DOWN - SOUTH - WEST - STRANGLE PIRATE - GET KEY - EAST - EAST - NORTH - UNLOCK DOOR WITH KEY - OPEN DOOR - DOWN - SOUTH - SOUTH - PUSH TRAP WEST - PUSH TRAP NORTH - PUSH TRAP NORTH - PUSH TRAP NORTH - SOUTH - SOUTH - WEST - GET HAMMER - CLOSE DOOR - LOCK DOOR WITH KEY - SOUTH - WEST - NORTH - NORTH - WEST - GET KNIFE - EAST - SOUTH - UP - WEST - SOUTH - WEST - ESAMINE PLANK - GET NAIL WITH HAMMER - DROP HAMMER AND RAGGED CLOTHES - EAST - EAST - NORTH - DOWN - SOUTH - EAST - NORTH - UNLOCK DOOR WITH KEY - OPEN DOOR - DOWN - WEST - NORTH - CUT RAT WITH KNIFE - GET DIAMOND IN

ISLAND OF LOST HOPE

RAT - SOUTH - SOUTH - WEST - GET ALL IN CRATE - EAST - EAST - NORTH - UP - CLOSE DOOR - LOCK DOOR WITH KEY - SOUTH - WEST - NORTH - UP - EAST - CLIMB DOWN - TIE STRING TO NAIL - BEND NAIL - CATCH FISH - WEST - DOWN - NORTH - GIVE FISH TO PIRATE - GET COIN - DROP STRING - DROP IRON KEY - DROP KNIFE - SOUTH - UP - SOUTH - SOUTH - OPEN DOOR - SOUTH - LOOK BEHIND PAINTING - GET KEY -

DEJA VU II

Ace Harding, l'intrepido investigatore di «Deja Vu», è ancora nei guai: dopo aver risolto il mistero della morte di Joey Siegel, egli si trova ora a dover rispondere della scomparsa di 112 mila dollari, che il gangster Tony Malone aveva consegnato a Siegel e che devono essere recuperati entro una settimana, trascorsa la quale Stogie Martin, il sicario di Malone, farà in modo che gli capiti qualche spiacevole incidente.
«AmigaByte» ha già pubblicato, sul fascicolo numero 7, la soluzione del primo episodio di «Deja Vu»: per aiutare Ace Harding, e quindi voi, a salvare la pelle anche in questa nuova avventura, presentiamo adesso la soluzione completa del secondo episodio di questa serie poliziesca della MindScape, intitolato «Deja Vu II: Lost in Las Vegas».
All'inizio del gioco vi trovate nel bagno di una camera dell'albergo-casinò Lucky Dice di Las Vegas, di proprietà di Tony Malone.

Prendete ed indossate la giacca ed i pantaloni; in una tasca troverete una chiave ed un portafoglio, contenente due ritagli di giornali, la vostra patente di guida, una banconota da 10 dollari ed una moneta da 10 centesimi.
Uscendo dal bagno incontrerete Stogie Martin, lo scagnozzo che Malone vi ha messo alle costole per sorvegliarvi.

PUNTARE AL CASINÓ

Raccogliete la fascetta del sigaro che lascia cadere a terra ed andate in camera da letto.
Prendete l'orario dei treni, uscite e, passando per il corridoio, entrate nell'atrio del casinó; cambiate la banconota in fiche alla cassa e recatevi al banco del black jack.

Il vostro vecchio amico Rudy Kowalski, lo stesso che appare su uno dei ritagli di giornale, lavora qui come croupier: cercate il suo tavolo e mostrategli il ritaglio in modo che si ricordi di voi. Vi riconoscerà e vi strizzerà l'occhio:

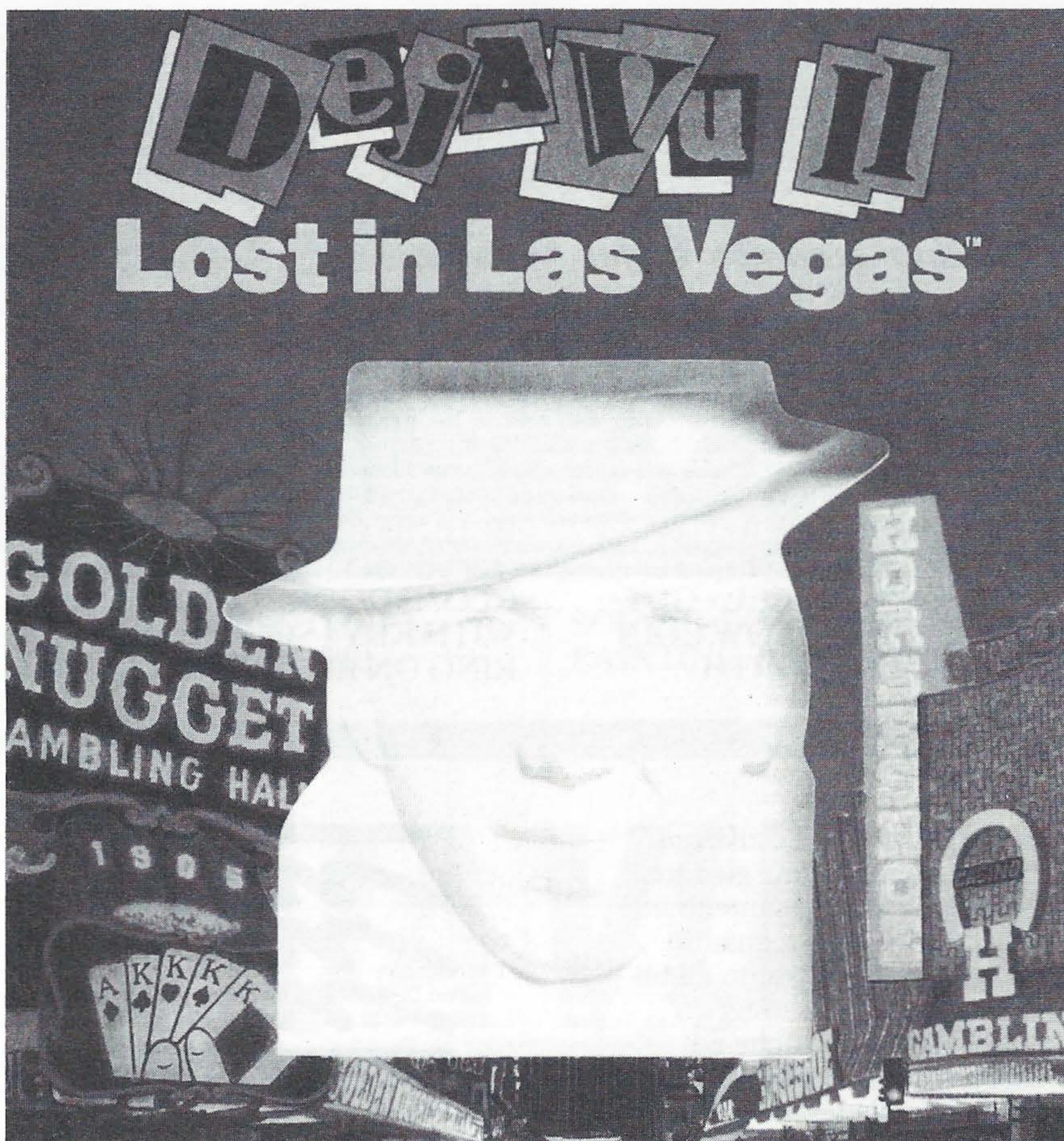
UNLOCK TRUNK WITH KEY - OPEN TRUNK - LOOK INTO TRUNK - GET CHEST - NORTH - WEST - PUT ALL IN BOAT BUT COIN - EAST - EAST - NORTH - GET BALL - SOUTH - PUT BALL IN BOAT - NORTH - GET BALL - SOUTH - PUT BALL IN BOAT - NORTH - GET BALL - SOUTH - PUT BALL IN BOAT - NORTH - NORTH - WEST - DOWN - NORTH - GET KNIFE - SOUTH - UP - WEST - NORTH - CUT NET WITH KNIFE - SOUTH - SOUTH - SOUTH - SOUTH - SOUTH - EAST - TURN WHEEL - GET RING - EAST - THROW COIN INTO WATER - NORTH - WEST - WEST - GET ALL IN BOAT BUT KEY - EAST - NORTH - UP - WAIT - LOOK - (ripetete i due comandi precedenti fino a quando non appare l'avvertimento che la nave è quasi giunta all'isola) - DOWN - SOUTH - WEST - ENTER BOAT - WEST - NORTH - THROW CHEST DOWN CLIFF - SOUTH - DIG GROUND - GET HORN - WEST - WEST - NORTH - BLOW HORN - NORTH - EAST - GET KEY IN CHEST - EAST - LOOK BEHIND BUSHES - UNLOCK DOOR WITH KEY - SOUTH - CLOSE DOOR - LOCK DOOR WITH KEY - SOUTH - PUT RING ON RIGHT BIN - GET RING - PUT RING ON RIGHT BIN - SOUTH - EXAMINE CHEST - READ INSCRIPTION - PUT DIAMOND IN HOLE - GET TORCH - NORTH - NORTH - UNLOCK DOOR WITH KEY - NORT - GET RING - NORTH - NORTH - WEST - DROP STRAW - BURN STRAW WITH TORCH - DROP KEY - GET BALL - WEST - SOUTH - SOUTH - EAST - EAST - EAST - BLOW HORN - CLIMB ON WHALE - NORTH - PUT BALL IN CANNON - FIRE CANNON - EAST - BURN CAPTAIN WITH TORCH - OPEN DOOR - SOUTH - UNITE WOMAN - GIVE RING TO WOMAN.

a questo punto, grazie alla sua collaborazione, dovete giocare e vincere rapidamente almeno una decina di fiche, fino a quando Rudy non verrà sostituito da un altro croupier.
Raccogliete tutte le fiche ed andate a cambiarle alla cassa, infilandole nella fessura.
Uscite ora dal casinó e dirigetevi verso ovest nel deserto, seguendo le tracce di pneumatici sulla strada. Arriverete fino ad una lavanderia, nella quale per il momento non potete entrare.
Andate verso est, tornando a Las Vegas, fino alla stazione ferroviaria. Sul tabellone degli orari localizzate il binario del primo treno in partenza per Chicago (deve apparire la dicitura «En Route» di fianco alla destinazione): recatevi al binario ad aspettare il treno, e salite in vettura.
Prima della partenza, pagate al controllore, i venti dollari del biglietto quando ve li chiede ed aspettate, dormendo, di arrivare a destinazione.

IN GIRO PER CHICAGO

Alla stazione di Chicago prendete la moneta da 25 centesimi e datela all'edicolante per comprare un giornale; leggete l'indirizzo dell'obitorio ed uscite dalla stazione, salendo sul primo taxi.
Il tassista è Gabby, una vostra

vecchia conoscenza: è sordo, e dovrete fargli leggere l'indirizzo di casa vostra sulla patente di guida per farvi trasportare a destinazione. Una volta giunti a casa, aprite la porta dell'appartamento: scoprirete che il locale è stato messo a soqquadro da Stogie Martin, come rivela la fascetta da sigaro sul pavimento.
Raccogliete dal cassetto a terra le due chiavi, la scatola di munizioni

ed il coltello a serramanico; poi prendete la pila dal pavimento, ed il revolver ed il denaro dalla giacca appesa alla parete. Caricate la 38 special con i proiettili della scatola e tornate nell'atrio.
Aprite la casella della posta con una delle due chiavi appena raccolte e prendete le tre lettere in essa contenute: leggetele e lasciatele sul posto, prima di uscire.
Risalite sul taxi e mostrate a Gabby l'altro ritaglio di giornale, sul quale è scritto l'indirizzo del bar di Joey Siegel (lo stesso in cui era ambientata la prima parte di «Deja Vu»).
Arrivati a destinazione, percorrete la strada laterale a sinistra e salite lungo la scala anti-incendio, spostando le assi che chiudono la finestra.
Entrate nell'ufficio di Joey e perquisite la scrivania, nella quale troverete una matita, e spostate il telefono. Raccogliete la chiave che l'apparecchio nascondeva e tornate sui vostri passi, recandovi ora nel cortile posteriore.
Aprite la porta con il coltello a serramanico ed entrate nel bar, accendendo la pila per riuscire a vederci.
Entrate nella taverna: se avete già giocato alla prima parte di «Deja Vu» saprete che dietro lo scaffale con le bottiglie c'è un passaggio segreto. Apritelo azionando l'unica bottiglia sulla destra, ed entrerete nella bisca clandestina.
Aprite la slot machine di destra con la chiave che avete raccolto poco prima nell'ufficio di Siegel, e prendete la tessera che si trova al suo interno. Prelevate anche l'agenda con la lista degli informatori ed uscite, tornando al taxi. Mostrate a Gabby la tessera

appena trovata e fatevi portare a casa di Sugar Shack.
Aprite la porta della cantina sparando sulla serratura ed entrate nell'appartamento: prendete le ricevute dal comodino della camera e confrontate le cifre con quelle dell'agenda degli informatori; scoprirete che Sugar Shack è l'informatrice contrassegnata con il numero zero.
Prendete l'uniforme da poliziotto dall'armadio ed indossatela, dopo esservi tolti i calzoni; esaminate bene l'interno dell'armadio, fino a trovare un aspirapolvere.
Tagliatene il sacchetto con il coltello e prendete la lettera che, come Sugar ha scritto, in caso di sua scomparsa deve essere consegnata alla polizia ma non mostrata ad un certo capitano Carlston, corrotto da Siegel.
Tornate al taxi e mostrate a Gabby il giornale sul quale è indicato l'indirizzo dell'obitorio.
Entratevi senza problemi, grazie all'uniforme che state indossando, e recatevi alle cella frigorifere.

UNA VISITA ALLA MORGUE

Nella seconda da destra troverete il cadavere di un certo Thomas Bondwell: tornate all'ingresso e chiedete alla guardia informazioni su questo Bondwell.
La guardia vi consegnerà una scatola contenente gli effetti personali del morto; prendete le chiavi e il portafoglio, che contiene dieci dollari, una patente di guida ed uno scontrino del deposito bagagli della stazione.
Uscite e salite ancora una volta sul taxi, dove mostrerete a Gabby la patente di Bondwell per farvi trasportare al suo appartamento.
Scoprirete però che la casa è bruciata a causa di un incendio di origine dolosa. Rimettete i vostri indumenti, mostrate quindi a Gabby l'orario ferroviario per farvi riportare alla stazione, e tornate a Las Vegas con il primo treno disponibile.
Al deposito bagagli consegnate all'addetto lo scontrino trovato nel portafoglio di Bondwell ed aprite la valigia che egli vi darà.

Controllate i vestiti: troverete una fotografia che raffigura Siegel mentre stringe la mano ad un uomo, contrassegnato dalle iniziali «D.V.», davanti alla lavanderia alla quale vi eravate già recati all'inizio.

IL POTERE NEL TERRITORIO

Leggete quindi la lettera di Bondwell a Malone: scoprirete che Bondwell ha dato a Joey Siegel parecchio denaro, informando di ogni pagamento il misterioso «D.V.», il quale cerca di allargare la propria influenza sul territorio di Malone a spese di quest'ultimo.
Prendete lettera e foto e restituite la valigia, tornando poi al vostro albergo.
Chiamate l'ascensore e salite al terzo piano: nel corridoio troverete un contenitore per la biancheria sporca. Apritelo ed entrateci, aspettando finché non viene richiuso e portato fino alla lavanderia.
Verrete scoperti da due killer, che vi perquisiranno dopo avervi ammenettato, e troveranno la lettera di Bondwell e la fotografia.
Liberatevi delle manette usando la cassa a sinistra del quadro e correte su per le scale, aprendo la porta esterna e scendendo di nuovo per rientrare nel contenitore della biancheria.
I killer, vedendo la porta aperta, crederanno che siate fuggiti da quella direzione: appena si saranno allontanati, salite nuovamente le scale ed aprite la porta interna entrando nell'ufficio.
Prendete la busta vuota indirizzata a Malone sulla scrivania, la chiave di ottone e la calamita contenuta nella scatola.

DOPPIO GIOCO

Uscite dalla lavanderia e tornate all'albergo, dove prenderete ancora l'ascensore: posizionate la calamita sopra il pulsante del quarto piano e salirete fino al quinto.
La porta di sinistra del pianerottolo conduce all'ufficio di Dan Ventini (il famigerato «D.V.»): sulla scrivania troverete un quaderno contenente alcuni pagamenti fatti a Malone, le cui somme non corrispondono però a quelle dell'agenda di Sugar.
Prendete poi dalla scrivania il fermacarte e spezzatelo, recuperando la freccia.
Tornate alla lavanderia, aprendo la porta scorrevole con la chiave di ottone e salendo fino all'ufficio.
Lanciate la freccia contro il bersaglio, facendo scattare un meccanismo che apre un passaggio segreto.
Entrate e recuperate dalla scrivania la lettera e la foto di Bondwell che vi erano state sottratte dai killer, oltre che un altro elenco di pagamenti.
Avete ora tutte le prove del complotto di Ventini contro Malone, ma dovete riuscire a convincere quest'ultimo ed a sbarazzarvi di entrambi.
Lasciate quindi nell'ufficio una delle fascette di sigaro per far credere a Ventini che è stato Stogie Martin, lo scagnozzo di Malone, a perquisire la sua scrivania ed a recuperare le prove.
Tornate poi all'albergo, salite al quinto piano ed entrate nell'ufficio di destra, quello di Malone, depositandovi le lettere di Sugar e di Bondwell e l'agenda di Siegel.

VERSO IL DESERTO

In questo modo avete messo i due gangster l'uno contro l'altro: scappate ora velocemente uscendo dall'albergo e dirigetevi verso est nel deserto. Se non avete commesso errori, verrete informati dopo qualche minuto che Malone è stato ucciso a colpi di pistola da Ventini e che quest'ultimo è morto poco dopo a causa di una bomba incendiaria precedentemente messa da Malone nella lavanderia.
Tornate ora alla stazione e prendete il treno per Chicago.
La vicenda è ora felicemente conclusa: i giornali riferiscono della morte di Malone e di Ventini e si congratulano con voi per la soluzione del caso. □

EXTEND 1.3

Altrove, già nel fascicolo di ottobre di AmigaByte, ci siamo occupati della gestione delle librerie con **AmigaBasic**, spiegando come per loro tramite sia possibile aggirare alcune delle limitazioni, spesso troppo vincolanti, di questo popolare linguaggio.

Proprio a coloro che non vogliono rinunciare alla semplicità ed immediatezza di AmigaBasic, ma che allo stesso tempo sono esasperati dalla sua scarsa versatilità, la Sun-Smile Software ha dedicato «**Extend 1.3**», una libreria contenente 72 nuove potenti funzioni.

L'installazione e l'uso della libreria «Extend» è molto semplice: basta copiare nella directory Libs del vostro disco di sistema i file «**Extend.library**» ed «**Extend.bmap**» fornito sul dischetto di «Extend 1.3». I due file insieme occupano meno di 35K; la loro installazione non dovrebbe perciò rappresentare un problema nemmeno per chi è sprovvisto di hard-disk. Se si desidera fare uso anche delle funzioni della libreria **ARP** (necessarie per l'uso di qualcuna delle routine fornite con «Extend») la stessa operazione andrà compiuta anche per i file «**Arp.library**» ed «**Arp.bmap**».

A questo punto la libreria sarà accessibile a tutti i programmi Basic, a patto che essi contengano all'inizio queste due righe:

Library «Extend.library»
SetWindow Window(7)

Il primo comando è necessario per aprire la libreria e renderla disponibile al programma; il secondo deve invece essere obbligatoriamente inserito all'inizio di tutti i listati che facciano uso della libreria Extend, come più volte specificato nelle istruzioni.

L'uso delle singole funzioni della libreria è molto semplice grazie ai numerosissimi esempi forniti già sotto forma di subroutine che, con pochissime modifiche, possono essere incorporate nei propri programmi.

I comandi e le relative subroutine della libreria Extend sono rivolti ovviamente a migliorare le prestazioni di AmigaBasic proprio sotto gli aspetti in cui questo linguaggio è più carente, come ad esempio la gestione dei requester o il supporto del formato IFF.

Tra le decine di funzioni disponibili, non mancano quelle di utilizzo generale (**Busy**, **Change**, **Credits**, **GoodBye**, **PassTIME**, **Protect**, **SetBit**, **TestBit**, **SetWindow**) o dedicate alla gestione della memoria (**FreeMem**, **Memory**, **BinLoad**, **BinSave**) e delle operazioni di input/output con AmigaDos (**DosCommands**). Grazie alla libreria Extend è possibile inoltre manipolare file in formato IFF sia dal punto di vista della grafica (**Xscroll**, **LoadIFF**, **MoveScreen**, **SaveIff**, **ViewHAM**) che da quello del sonoro (**Channel**, **ExtendedSound**, **SetAddress**, **SetLength**, **SetRate**, **SetVolume**, **SoundOff**, **StartSound**, **StopSound**).

Non mancano routine di gestione dei menu (**CheckSub**, **CommandKey**, **Mat**, **SetSub**, **SubColor**, **SubMenu**), dei requester (**ArpRequester**, **GetFiles**, **ReqCOLOR**, **REQUESTer**, **SReqColor**, **StringREQUESTer**) e dei font bitmap (**Font**, **SetStyle**).
Grande risalto è dato infine alla gestione dei gadget, sia in senso generale (**Check**, **GadCOLOR**, **GadgetBorderColors**, **GadgetsOFF**, **GadgetsON**, **GadgetTextCOLOR**, **GadRepeat**, **GRelBottom**, **GRelRight**, **OffGADget**, **ONGADget**, **Refresh**, **RemoveGADgets**, **SetHilight**) che specificamente per quelli grafici (**AlternateImages**, **BoolGadgetImage**, **PictureGadget**, **PlanePick**, **SetImage**), quelli proporzionali (**PropGadget**, **PropMove**, **Proportionalgadget**, **PropRead**), quelli booleani (**Gadget**, **WaitBooleanGadget**), ed infine quelli per l'input di stringhe (**ClearBuffer**, **GetInput**, **GetString**, **PutString**, **Status**, **WaitGad**).

I nomi stessi dei comandi sono pressoché auto-esplicativi per chiunque abbia un minimo di cono

scenza dell'inglese o di linguaggi di programmazione: per far eseguire un reset del computer dall'interno di un programma Basic, ad esempio, basta usare il comando **GoodBye**.

Anche la sintassi è molto semplice: per visualizzare un'immagine grafica in modo Ham, è sufficiente ricorrere al comando **ViewHam SADD ("nomefile"+chr$(0))**; per cambiare l'intestazione di una finestra, si usa il comando **Change SADD ("Nuovo Titolo"+chr$(0))**, etc.

Il parametro SADD definisce il testo da passare come argomento ai comandi, seguito da un carattere Ascii 0 (una forma usata spesso in linguaggi come il C per delimitare il termine di una stringa).

«Extend 1.3» è un prodotto dav-

vero interessante per chiunque programmi in Basic. La software house assicura nella (scarna) documentazione del programma la compatibilità della libreria anche con il compilatore «**A/C Basic**» della Absoft o con l'interprete/compilatore «**Hisoft Basic**»; persino il «**GFA Basic**» è interfacciabile con la libreria, grazie ad un'utility compresa nel pacchetto, anche se la maggiore potenza di questo interprete dovrebbe rendere superflua l'aggiunta delle funzioni di «Extend».

SunSmile Software
533 Fargo Avenue
Buffalo, NY 14213
USA

ART DEPARTMENT

Il nome di questo nuovo programma della ASDG Inc. (la stessa del «Cygnus Editor» e di «Professional ScanLab») è ingannevole, non trattandosi infatti di un programma di grafica pittorica o comunque dedicato alla creazione di disegni od immagini.

«The Art Department 1.01» (o «TAD», come viene chiamato per brevità anche nella documentazione) appartiene invece alla categoria delle utility dedicate all'**«image processing»**, della quale gli esponenti più noti sono «Pixmate» e «Butcher».

È della gestione di immagini pre-esistenti, e non perciò della loro creazione ex-novo, che «TAD» si occupa; la differenza tra «TAD» e gli altri programmi sopracitati risiede nel fatto che questi ultimi sono più orientati verso l'elaborazione e l'aggiunta di particolari effetti visivi alle immagini, mentre il programma della ASDG si occupa prevalentemente della conversione tra formati differenti.

Il vero punto di forza di «TAD» risiede infatti in quelli che la documentazione chiama «loader»: moduli aggiuntivi (molti dei quali venduti separatamente dal programma principale) che consentono il caricamento e la gestione di svariati formati grafici, alcuni dei quali non impiegati in ambito Amiga ma molto diffusi su altri sistemi operativi.

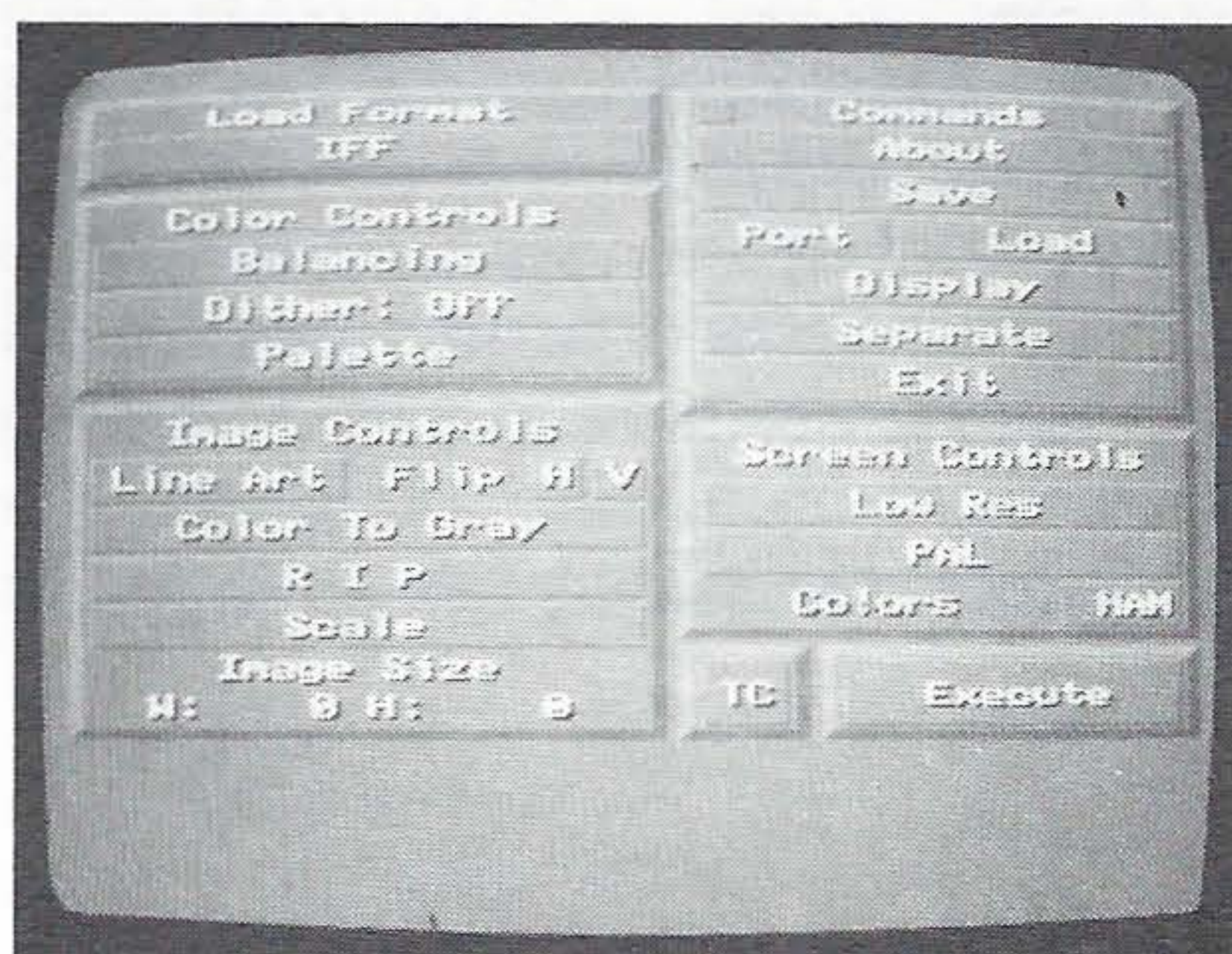
I moduli forniti con il disco di «TAD» sono relativi ai formati **IFF** e **DV21**; quelli opzionali gestiscono i formati **GIF**, **Sculpt**, **Silver**, **Rendition**, **Dpaint II Enhanced** (MsDos) e **Targa** (versioni 1, 2, 9 e 10). La ASDG promette comunque che distribuirà regolarmente nuovi moduli per eventuali formati aggiuntivi.

L'aspetto più interessante di «TAD» risiede nella comodità della sua interfaccia utente: tutte le opzioni sono accessibili tramite un pannello di controllo (che rappresenta l'unico schermo del programma) molto funzionale ed intuitivo.

Il primo gruppo di opzioni (**Load Format**) è relativo alla scelta del formato del file da caricare: le scelte disponibili dipendono dai moduli presenti sul disco del programma. Il secondo gruppo (**Color Control**) permette elaborazioni cromatiche delle immagini: le tre opzioni disponibili sono **Palette** (per cambiare manualmente i colori), **Balancing** (agisce sui rapporti tra le sfumature di colore) e **Dithering** (per scegliere uno tra i sei algoritmi di dithering, o retinatura, disponibili).

Segue un gruppo di opzioni (**Image Control**) relative a semplici funzioni di manipolazione dell'immagine: **LineArt** produce una copia dell'immagine evidenziandone solo i contorni (immaginate di ricalcare su di un foglio di carta i tratti essenziali di una fotografia); **Flip** ruota specularmente l'immagine sugli assi orizzontale o verticale; **Color to Gray** converte un'immagine a colori in una in bianco e nero, con sedici sfumature di grigio; **RIP**, nonostante il nome funereo, è l'abbreviazione di «Remove Isolated Pixels» e si occupa appunto di far scomparire da un'immagine i pixel di colore isolati (utile specialmente in modo Ham); **Scale** infine si occupa di variare il rapporto di scala tra gli assi dello schermo, consentendo di allungare o comprimere a piacere il disegno.

I comandi del gruppo **Commands** sono invece relativi principalmente alle operazioni di input/output dei file: **Load** e **Save** richiamano il requester per la gestione dei file su disco; il quadro a destra di Load può essere clickato per selezionare il modo di visualizzazione **Portrait** o **Landscape** (orizzontale o verticale); **Separate** permette appunto operazioni di separazione dei colori; **Display** visualizza l'immagine caricata; **About** fornisce, come di consueto, informazioni sulla release e sugli autori del programma, mentre **Exit** ne termina l'esecuzione.

L'ultimo gruppo di opzioni è denominato **Screen Controls** e comprende tre riquadri sui quali si può

clickare per selezionare rispettivamente la risoluzione dell'immagine, quella dello schermo (Pal o Ntsc) ed il numero di colori. Inutile dire che tutti i modi grafici di Amiga sono pienamente supportati, compreso il nuovo Dynamic HiRes (4096 colori in alta risoluzione) introdotto dalla NewTek.

Il gadget **Execute** nell'angolo inferiore destro dello schermo serve infine per far rielaborare a «TAD» l'immagine in memoria dopo averne variato qualche parametro tramite gli altri menu.

«TAD» è in definitiva un'eccellente programma, i cui pregi superano abbondantemente le lievissime idiosincrasie: la più fastidiosa è relativa al comando Exit, che non chiede conferma all'utente dell'intenzione di uscire dal programma e che quindi può facilmente essere clickato per errore con risultati disastrosi.

Come «image processor» «TAD» non è in grado, né si prefigge, di competere con l'attuale concorrenza: ma è la versatilità nella gestione dei formati grafici a renderlo, per il momento, unico nel suo genere.

ASDG Inc.
925 Stewart Street
Madison, Wisconsin
53713 USA

T.A.C.L.

Chi ha mai detto che tutti i linguaggi di programmazione sono per loro natura complicati, astrusi o noiosi? Se quest'affermazione può risultare veritiera per molti di essi, esistono comunque delle eccezioni alla regola, come quella rappresentata da «**The Adventure Construction Language**» («T.A.C.L.»), prodotto dalla Micro Momentum.

«T.A.C.L.» è un linguaggio in piena regola, con tanto di compilatore e debugger, ma non lo si può certo definire complesso né tantomeno noioso: il suo compito è, al contrario, quello di divertire, essendo dedicato alla programmazione di giochi «adventure».

Il pacchetto del programma comprende due dischetti ed un semplice manuale (in inglese) molto chiaro ed esauriente anche per chi non ha troppa dimestichezza con compilatori ed affini. Il primo dischetto contiene «**Madv**» (Make Adventure), ovvero il compilatore vero e proprio; «**Vged**» (Vector Graphics Editor), un semplice programma per la creazione di disegni vettoriali; e «**Padv**» (Play Adventure), il programma per l'esecuzione delle avventure create con «Madv». Il secondo disco contiene invece due giochi dimostrativi completi di sorgente.

Le avventure generate da «T.A.C.L.» sono di tipo tradizionale; quelle cioè in cui il giocatore impartisce al programma le proprie direttive esclusivamente tramite comandi scritti sulla tastiera.

Il «player» è liberamente distribuibile, in modo che l'utente possa tranquillamente far circolare o commercializzare il frutto delle proprie fatiche senza incorrere in problemi di violazione di copyright.

La Micro Momentum assicura che questa implementazione del linguaggio «T.A.C.L.» è identica a quella delle imminenti versioni Ibm e Macintosh: un'avventura creata su Amiga può perciò essere convertita per girare su altri computer senza particolari difficoltà.

«T.A.C.L.» è un linguaggio molto semplice da imparare: i comandi sono tutti chiari e si riferiscono alle tipiche azioni che possono essere intraprese da un giocatore in questo genere di simulazione (**Go**, **Grab**, **Drop**, **Move**, **Die**, **Object**, **Room**, **Action**, etc.). Come prevedibile, la logica del linguaggio è basata essenzialmente sui comandi **If-Then-Else-Endif** per determinare quali routine lanciare o quali immagini visualizzare se il giocatore compie o meno determinate azioni.

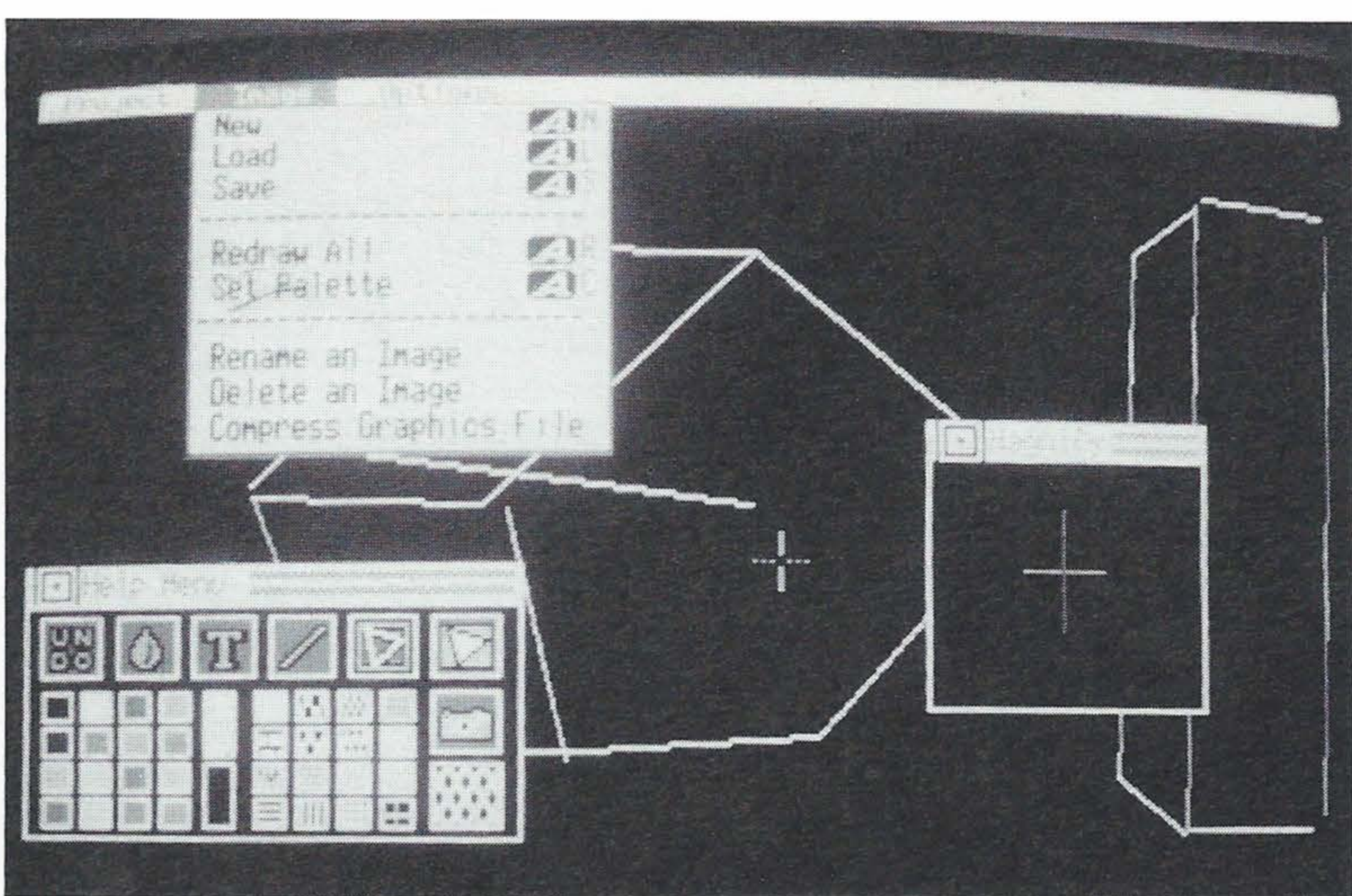
Il tipo di strutture e di approccio alla programmazione di un'adventure è stato discusso nei dettagli sui fascicoli 19 e 20 di AmigaByte. Gli articoli in questione si riferivano all'uso del linguaggio AmigaBasic, ma gran parte delle tecniche di progettazione dell'avventura rimangono valide anche operando con «T.A.C.L.».

Il linguaggio prevede l'uso di due tipi di grafica: immagini nel classico formato IFF (che hanno però lo svantaggio di occupare normalmente una discreta quantità di spazio su disco) o disegni realizzati con il programma «**Vged**» incluso. Quest'ultimo sfrutta una tecnica vettoriale che memorizza solo le coordinate dei punti che uniscono le linee componenti del disegno. In questo modo il disegno è decisamente meno gradevole esteticamente di una tradizionale immagine IFF, ma occupa pochissimo spazio su disco.

Il linguaggio «T.A.C.L.», essendo stato prodotto per il mercato americano, è essenzialmente basato sull'uso di strutture grammaticali della lingua inglese: le preposizioni e le congiunzioni che ad esempio il programma riconosce sono inglesi (**and**, **in**, **for**, **with**, etc.). Con qualche adattamento e rinunciando a qualche sofisticazione è comunque possibile scrivere anche avventure che utilizzino la lingua italiana.

Tra tutti i programmi dedicati alla creazione di giochi ed adventure grafiche, questo è sicuramente uno tra i più validi; a suo sfavore gioca soltanto il fatto che il compilatore è utilizzabile soltanto da CLI e che l'utente deve impiegare un editor esterno per la stesura dei sorgenti, cose che rendono l'approccio più ostico soprattutto ai principianti.

Micro Momentum Inc.
P.O. Box 372
Washington Depot,
CT 06794 USA

SEGUE DA PAG. 19

limiti che ci auguriamo la HiSoft voglia superare nelle versioni future. Infatti, ben si sa quanto siano importanti le strutture tipo-C nella programmazione di Amiga; un Basic moderno e potente dovrebbe dare la possibilità di accedervi facilmente, magari attraverso un tipo **RECORD**, come già accade nel **Fast basic**, senza costringere come fa «HiSoft Basic», il programmatore a una serie di contorti (e impossibili da «leggere») **PEEK** e **POKE** per trovare il giusto indirizzo di memoria dei diversi membri della struttura. Inoltre, c'è la possibilità di definire delle costanti; per esempio:

CONST SI=1, NO=0

significa che tutte le volte che scriverò **NO** nel mio listato sarebbe come scrivessi **0** e **SI** come scrivessi **−1**; questa tecnica aumenta la chiarezza e la leggibilità del programma, ma presto ci accorgiamo che è inutilizzabile quando si vogliano definire delle costanti con valori non interi, magari di tipo stringa o a virgola mobile.

COMPILATORE E INTERPRETE

Esiste inoltre una nutrita serie di casi in cui il nostro compilatore e l'interprete AmigaBASIC differiscono leggermente, o perché si sono voluti migliorare alcuni comandi, o perché in qualche caso non si è ancora ottenuta la totale compatibilità od il perfetto funzionamento. Per quanto riguarda la produzione del suono abbiamo provato a compilare il listato «music», contenuto nel dischetto *Extras*, directory *Basicdemos*, cogliendo un clamoroso insuccesso. Un'altra caratteristica che può risultare fastidiosa è, per esempio, il parzialmente diverso comportamento dei comandi grafici **GET** e **PUT**, che emerge solo in particolari casi, in cui forse il programmatore si è comportato in modo non troppo ortodosso, ma che ci sono sembrati difficili da rimediare; sempre a proposito di grafica c'è da segnalare (in positivo) la possibilità di aggiungere al comando **COLOR** un terzo parametro indicante il modo di tracciamento (**JAM1, JAM2, COMPLEMENT, INVERSVID**), così da evitare l'apertura della libreria grafica solo per poter invocare la funzione apposita. Anche la gestione dello schermo è stata migliorata con lo stesso sistema che conserva la compatibilità, cioè con l'aggiunta di un ulteriore possibile parametro al comando **SCREEN**, parametro nel quale deve essere inserito il **ViewMode**, il modo di visualizzazione dello schermo.

Se il comando

SCREEN 2,320,256,5,1

apre uno schermo a 32 colori e bassa risoluzione, basta modificare di poco la sintassi:

SCREEN 2,320,256,6,1,&H800

per avere finalmente, anche da Basic e con grande facilità, uno schermo HAM in bassa risoluzione con 4096 colori!

Per quanto riguarda l'apertura delle finestre non è stato aggiunto nessun parametro al comando **WINDOW**, ma è stata invece ampliata la gamma di possibilità offerte dal penultimo parametro, che indica il tipo di finestra. Tra l'altro si ha ora la possibilità di creare una finestra senza bordi semplicemente sommando 128 al parametro di tipo. Per averne un esempio basta compilare e far girare il seguente «programma»:

WINDOW 2,,,128,−1

Anche la gestione delle librerie ci ha lasciato nel complesso più che soddisfatti. Occorre però fare attenzione a mettere il comando LIBRARY **prima** della dichiarazione delle varie funzioni. È noto che l'AmigaBASIC ha bisogno, per gestire le librerie, dei files «**.bmap**», dove sono memorizzati gli indirizzi delle varie funzioni ed i registri-macchina da usare per i parametri. Un aspetto particolarmente piacevole è che nel nostro caso solo il compilatore ha bisogno di questi file; il programma eseguibile invece potrà farne tranquillamente a meno, risparmiando in tempo di caricamento ed eventualmente anche in spazio su disco, e scaricandoci dalla necessità di copiare anche i file «.bmap» ogni volta che dobbiamo copiare il nostro programma compilato.

LA LIBRERIA HISOFT

Un caso particolare è dato dalla libreria **hisoftbasic.library**, che viene aperta ed utilizzata automaticamente (se abbiamo settato l'opzione **L**), ma che contiene due funzioni (**SafeAlloc** di allocazione e **SafeFree** di rilascio della memoria) che possiamo richiamare per conto nostro, dopo aver debitamente aperto la libreria. La sintassi è molto vicina a quella delle funzioni della exec.library **AllocMem** e **FreeMem**, alle quali rimandiamo, ricordando che l'unica differenza consiste nel fatto che SafeFree richiede come parametro solo il puntatore all'area di memoria; c'è però il grande vantaggio che se ci dimentichiamo di liberare la memoria, questa verrà comunque rilansciata per conto nostro all'uscita del programma. Il manuale contiene anche scarne indicazioni su come utilizzare da C o da assembler qualche altra funzione di questa potente libreria; ricordate però che il programma principale deve comunque essere scritto in Basic; è possibile inserire in un programma in Basic un «pezzo» in C, ma non è possibile l'operazione contraria!

IL COMANDO RESUME

Un elemento che purtroppo è stato trascurato e che ci ha lasciati insoddisfatti è l'implementazione del comando **RESUME** che, come è noto, gestisce il ritorno del programma da una routine di intercettamento degli errori; a volte, se la sintassi non è più che corretta, il compilatore genera un messaggio di errore ed inoltre, come risulta anche dal manuale di istruzioni, il compilatore non riconosce la forma **RESUME NEXT** (=continua come se niente fosse con l'istruzione successiva); può darsi che la programmazione strutturata possa fare a meno del vecchio RESUME NEXT, ma come la mettiamo con la compatibilità con i programmi già scritti in Basic?

In conclusione diremo che il pacchetto ha delle solide ed ottime basi e che si presenta quale strumento molto utile sia al programmatore alle prime armi, per la sua facilità di utilizzo e per la compatibilità con AmigaBASIC, sia al programmatore esperto, per l'ampio ventaglio delle sue possibilità e le estensioni introdotte. A volte si sente la mancanza di alcuni importanti dettagli che vorremmo implementati, altre siamo tormentati da qualche fastidiosi **bug** ma, essendoci noto come in passato la HiSoft ha migliorato costantemente i suoi prodotti, e considerando che quella che abbiamo esaminato è solo la versione 1.05, non resta che attendere e sperare che i nostri desideri vengano esauditi nelle versioni future.

Fai da te la tua Startup-sequence

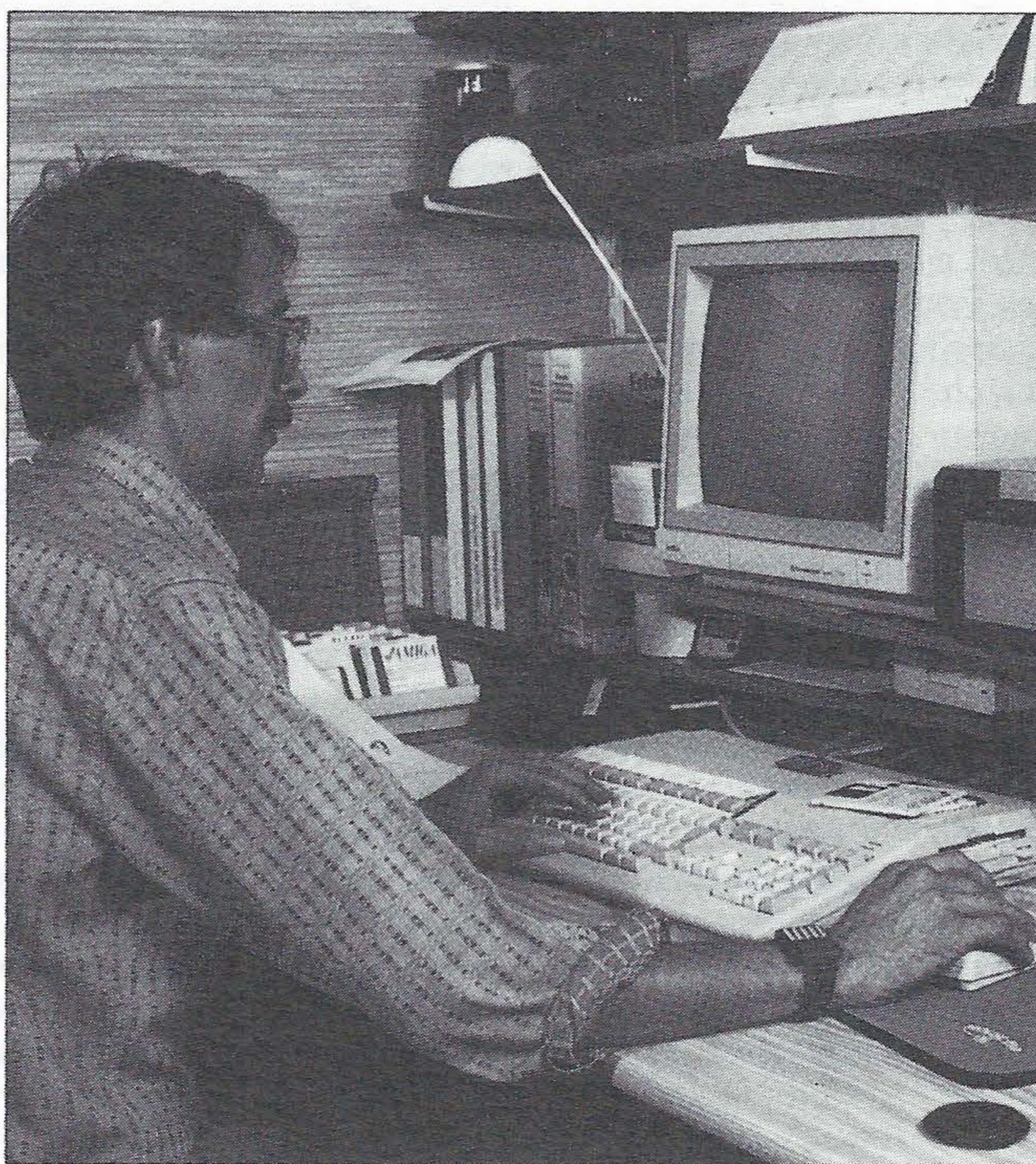
La Startup-sequence, l'editor Ed, la redirezione dell'output, il path di ricerca: tutti argomenti apparentemente complessi ma in realtà semplicissimi da comprendere.

di VITTORIO FERRAGUTI

Startup-sequence: cos'è? A che serve? Come la si può personalizzare? Queste sono le domande più comuni rivolte da coloro che muovono i primi passi nell'apparentemente complesso mondo di **AmigaDos**. Nonostante l'alone di mistero che spesso sembra circondarla, la Startup-sequence è uno strumento utilissimo e facile da usare per ricavare il massimo delle prestazioni dal proprio computer ed adattarlo alle proprie esigenze.

Il termine «**Startup-sequence**» si può rendere in italiano con l'espressione «sequenza di inizializzazione», ed in effetti proprio di questo si tratta: una serie di comandi che il sistema operativo esegue al momento dell'accensione o del reset del computer, per configurare il sistema e lanciare uno o più programmi.

COS'È LA STARTUP-SEQUENCE

La Startup-sequence è un file di testo in **formato Ascii** (ovvero composto da soli caratteri alfanumerici, senza nessun codice o carattere di controllo particolare) che normalmente si trova nella **directory «S»** dei dischetti.

Durante la fase del «**boot**» del computer (ovvero, la partenza del sistema dopo un reset o all'accensione), il sistema operativo va a cercare la Startup-sequence, la legge, e provvede ad eseguire uno per uno tutti i comandi in essa contenuti, siano essi comandi AmigaDos o programmi da lanciare.

Per spiegare il meccanismo di modifica e di personalizzazione della Startup-sequence, faremo ricorso ad un esempio pratico di sicuro interesse per gli utenti Amiga in possesso di un solo drive: illustreremo infatti per filo e per segno come fare per caricare in memoria i principali comandi di AmigaDos, evitando la necessità di mantenere sempre inserito il disco **WorkBench** nel drive interno della macchina durante il normale utilizzo del computer.

Per questo esperimento, come del resto in generale, vale la regola di operare solo su di una copia del disco WorkBench e non sull'originale, anche per evitare danni irrimediabili al disco in caso di errori involontari o di malfunzionamenti.

La procedura per la copia di un dischetto è descritta abbastanza dettagliatamente anche nel manuale di introduzione di Amiga: in breve, in ambiente WorkBench si può ricorrere all'opzione «**Duplicate**» del **menu «WorkBench»**, dopo aver selezionato l'icona del disco da copiare; in ambiente CLI, si può invece usare il comando «**DiskCopy df0: to df0:**" (il discorso vale sempre per chi possiede un solo drive).

L'EDITOR ED

Trattandosi di un file di testo, la Startup-Sequence è modificabile per mezzo di un qualsiasi text-editor, cioè di un programma per la stesura e la modifica di testi in formato Ascii.

Gli utenti più esperti saranno certamente in pos-

sesso di uno dei numerosi editor distribuiti commercialmente. quali «**TxEd**» o «**Cygnus Editor**», o di qualche analogo programma di pubblico dominio, come ad esempio «**Uedit**» o «**Qed**». In loro assenza, si può fare ricorso all'editor standard del WorkBench, cioè al comando di AmigaDos «**Ed**».

AmigaByte si è occupata in dettaglio del funzionamento dell'editor «Ed» sulle pagine del fascicolo 23, elencando e spiegando le funzioni di tutti i suoi comandi; in questa occasione, quindi, ci limiteremo a descrivere solo i comandi essenziali relativi al caricamento ed al salvataggio di testi.

Passiamo alla pratica: caricate normalmente il WorkBench, inserendolo nel computer dopo l'accensione ed attendendo fino a quando non appare sullo schermo il tradizionale ambiente ad icone. Clickate due volte sull'icona del disco WorkBench e poi due volte su quella del **cassetto System** in esso contenuto.

La finestra del cassetto System contiene a sua volta un'icona denominata «**Cli**» o «**Shell**» (dipende dalla versione di WorkBench che state utilizzando); è comunque indifferente l'utilizzo dell'uno o dell'altro ambiente, perciò clickate su quella disponibile ed attendete fino a quando non appare la finestra con il «**prompt**» del sistema operativo.

Il termine «prompt» indica quel carattere (normalmente il simbolo «>») che precede il cursore e segnala che il computer è pronto all'accettazione di comandi.

Al prompt digitate dunque questo comando:
Ed s:startup-sequence

Il drive entrerà in azione e, dopo qualche secondo, apparirà una finestra contenente diverse righe di comandi AmigaDos: è la Startup-sequence del dischetto WorkBench, pronta per essere modificata (in gergo «editata») secondo i vostri desideri.

Due parole sull'editor «Ed»: oltre che poter digitare direttamente il testo e spostare il cursore tramite le frecce, l'editor accetta alcuni comandi particolari per svolgere determinate funzioni. Quelle che ora interessano a noi sono la funzione di salvataggio del testo modificato, e quella di abbandono del testo senza modifiche: la prima serve per memorizzare la Startup-sequence una volta inserite le variazioni desiderate; la seconda potete usarla nel caso vi rendiate conto di aver commesso errori irreparabili e vogliate uscire per ricominciare tutto dall'inizio.

Per salvare il testo ed uscire si preme il tasto **Esc** e, alla comparsa dell'asterisco nell'ultima riga della finestra, si digita il carattere «X» seguito da **Return**; per uscire senza salvare invece, il carattere da digitare dopo la pressione di **Esc** sarà la lettera «**Q**» (per «Quit», in inglese «interruzione») sempre seguita da **Return**.

L'ultimo comando che serve ai nostri scopi è quello di cancellazione di una riga, attivabile premendo insieme i tasti «**Control**» e «**B**».

Mettetelo subito in pratica: posizionate il cursore sulla prima riga del testo, fatevi coraggio e tenete premuto «**Ctrl-B**» fino a quando tutto il contenuto della Startup-sequence non è stato cancellato dallo schermo. Capite ora perché è indispensabile lavorare su di una copia del disco WorkBench?

Ora che la Startup-sequence è tabula rasa, potete cominciare a personalizzarla secondo le vostre esigenze; ma prima, occorre una piccola spiegazione sul metodo di esecuzione dei comandi AmigaDos.

COMANDI RESIDENTI

Una caratteristica particolare di AmigaDos, rispetto ad altri sistemi operativi quali **MsDos**, consiste nell'assenza di comandi o funzioni residenti: tutti i comandi sono esterni, cioè risiedono sul disco del sistema operativo sotto forma di singoli file eseguibili, e devono essere caricati in memoria dal dischetto ogni volta che è necessario eseguirli.

Il vantaggio principale di questo approccio consiste nel risparmio di memoria: ogni comando viene caricato solo quando è effettivamente necessario usarlo, e non porta via memoria preziosa risiedendo in Ram anche quando non lo si deve utilizzare; inoltre, il fatto di avere file separati per ogni comando rende più facile la loro sostituzione o modifica in caso di uscita di versioni più aggiornate o migliorate.

Sfortunatamente questo approccio comporta anche un inconveniente molto scomodo, specialmente per chi possiede un solo drive: la necessità di reinse-

```
Startup-sequence
; Startup-sequence per rendere residenti in memoria i piu' importanti
; comandi AmigaDos. Le righe precedute da un punto e virgola sono commenti
; e vengono ignorate in fase di esecuzione

MakeDir ram:c
; Crea una directory C in Ram.

copy c:dir ram:c
copy c:run ram:c
copy c:cd ram:c
Copy c:copy ram:c
copy c:delete ram:c
copy c:type ram:c
copy c:rename ram:c
; Copia i file contenenti i comandi AmigaDos desiderati dalla directory C
; del disco WorkBench alla directory C del disco Ram appena creata.

path ram:c add
; Inserisce la directory Ram:C nel percorso di ricerca dei comandi.

Sys:system/SetMap I
; Carica la configurazione della tastiera italiana. Usate "Setmap USA1" se
; avete una tastiera americana.

LoadWb
; Carica il WorkBench.

EndCli >Nil:
; Chiude la finestra CLI reindirizzando l'output al device NIL:
```

Nel nostro esempio la Startup-sequence carica in memoria i comandi AmigaDos di utilizzo più frequente.

```
Ed 1.14
Addbuffers df0: 10
c:SetPatch >NIL: ;patch system functions
cd c:
echo "A500/A2000 I Workbench disk.  Release 1.3 version 34.20*N"
Sys:System/FastMemFirst ; move C00000 memory to last in list
BindDrivers
SetClock load ;load system time from real time clock (A1000 owners should
              ;replace the SetClock load with Date)
FF >NIL: -0 ;speed up Text
resident CLI L:Shell-Seg SYSTEM pure add; activate Shell
resident c:Execute pure
mount newcon:
;
failat 11
run execute s:StartupII ;This lets resident be used for rest of script
wait >NIL: 5 mins ;wait for StartupII to complete (will signal when done)
;
SYS:System/SetMap i ;Activate the ()/* on keypad
path ram: c: sys:utilities sys:system s: sys:prefs add ; set path
                                                        ; for Workbench
LoadWB delay  ;wait for inhibit to end before continuing
endcli >NIL:
End of file
```

La normale Startup-sequence del disco WorkBench 1.3, caricata nell'editor Ed.

rire il dischetto WorkBench ogni volta che si deve eseguire un comando AmigaDos, anche solo per visualizzare la directory di un disco.

Esempio pratico: volete visualizzare la directory di un disco chiamato «AmigaByte_26» e possedete un solo drive? Dovete caricare il WorkBench, accedere al Cli, come spiegato precedentemente, e digitare il comando:

dir AmigaByte_26:

Il comando «**dir**» verrà caricato in memoria e vi verrà chiesto di inserire il disco «AmigaByte_26» nel drive; una volta inserito, potrete vederne il contenuto. Se, a questo punto, voleste ad esempio spostarvi in una directory di quel disco chiamata «**prova**», dovreste digitare:

cd AmigaByte_26:prova

A questo punto vi verrebbe chiesto di reinserire prima il disco WorkBench, per consentire il caricamento del comando «cd» in memoria, e poi di reinserire nuovamente il disco «AmigaByte_26» per permettere al comando in questione lo spostamento nella directory indicata.

Come vedete, si tratta di una procedura decisamente scomoda e complessa, che diventa poi insopportabile qualora le operazioni da compiere siano più complicate o numerose di quelle citate come esempio.

Per questo motivo, se non si vuole acquistare un secondo drive (o non ce lo si può permettere), è utile caricare e mantenere in memoria almeno i comandi principali di AmigaDos, in modo da evitare l'insorgere di quella sindrome nota a molti utenti Amiga come «gomito del drive singolo», che si manifesta in un indolenzimento del braccio usato per il continuo inserimento e la rimozione del disco WorkBench da parte di chi non possiede un drive esterno.

IL PERCORSO DI RICERCA

Torniamo alla nostra Startup-Sequence: digitate, nella prima riga dello schermo dell'editor, questi comandi:
Makedir ram:c
Copy c:dir ram:c
Cosa avete fatto? Con il comando «**Makedir**» avete detto di creare in **Ram**: (ovvero nel disco ram) una directory chiamata «C», mentre con il **comando «Copy»** avete fatto in modo da far copiare il file del **comando «Dir»** dalla **directory «C:»** del disco di sistema alla **directory «C»** appena creata del **disco Ram:**.

Ricordate (cfr. AmigaByte 24) che la directory «C:» del disco di sistema è appunto quella in cui risiedono normalmente (ed in cui vengono quindi cercati in caso di caricamento) i comandi principali di AmigaDos.

Potete ripetere l'ultima riga in modo da far copiare in Ram anche gli altri comandi AmigaDos che ritenete utile mantenere residenti in memoria. Ad esempio:
Copy c:cd ram:c
Copy c:copy ram:c
Copy c:delete ram:c
etc.

Per un elenco dei principali comandi AmigaDos, in ordine di importanza, fate sempre riferimento alle pagine di questa rubrica apparse sul fascicolo 24 di AmigaByte.

Una volta terminato l'inserimento delle righe per la copia dei comandi più utili in memoria, digitate questa riga:
Path ram:c add
Il **comando «path»** (in inglese «percorso») serve per comunicare ad AmigaDos appunto il percorso da seguire durante la ricerca di un comando da eseguire. Il percorso **di default** (cioè quello preimpostato, usato se non vengono inserite variazioni) è, come già detto, la directory «C» del disco di sistema: digitando cioè un qualsiasi comando al prompt del CLI, esso verrà prima cercato all'interno della directory corrente poi, se non viene in essa trovato, nella directory «C» di sistema (che normalmente contiene appunto i comandi AmigaDos). Se non viene trovato neanche lì, apparirà un messaggio di errore per informare che il comando è sconosciuto («Unknown command»).

Con il comando «path» questo percorso è modificabile: in particolare, con la riga sopracitata, il sistema operativo viene informato che al percorso di default è stata aggiunta (in inglese «**add**», appunto) un'altra directory: **Ram:c**.

Perciò gli eventuali comandi digitali al prompt verranno cercati anche in quella directory, prima che appaia il messaggio di errore: in questo modo sarà possibile togliere il disco WorkBench, mantenendo tuttavia la possibilità che i comandi, che sono stati precedentemente copiati in ram, siano ugualmente rintracciabili ed eseguibili dal sistema operativo.

IL CARICAMENTO DI WORKBENCH

Terminate l'editing della Startup-sequence inserendo le altre righe indispensabili al caricamento del

WorkBench ed alla configurazione del sistema:
sys:system/SetMap I
LoadWb
Endcli >nil:

La prima riga può sembrare strana: essa contiene il comando «**SetMap**», che carica la configurazione della tastiera per adattarla alle versioni nazionalizzate. Nel nostro esempio si fa riferimento alla tastiera italiana («I» appunto); se possedete una tastiera americana, sostituite il parametro «**Usa1**» a «I».

Davanti a «SetMap» è stato premesso il path necessario per caricarlo: «SetMap» è infatti uno tra i pochi comandi che non si trovano nella directory «C;», bensì nella **directory «System»** (insieme a «**Format**», a «**DiskCopy**», ed a qualche altro). Dal momento che il comando «Path» precedentemente inserito non fa menzione della directory «System», è necessario dire al sistema operativo dove trovare il file «SetMap» per poterlo eseguire, e glielo si comunica appunto premettendolo al nome del comando stesso.

Avremmo potuto adottare soluzioni diverse: si sarebbe potuto copiare anche il comando «SetMap» in memoria, inserendo nelle righe precedenti anche questa:
Copy sys:system/SetMap ram:c

Oppure, sarebbe stato possibile includere anche la directory «System» nel percorso di ricerca, modificando la riga del comando «Path» in questo modo:

Path ram:c sys:system add

Abbiamo invece scelto l'altra soluzione perché è utile per mostrare come la regola generale più valida consista nell'indicare sempre nella Startup-Sequence, insieme ai nomi dei programmi da caricare, anche il disco e la directory in cui si trovano: questo accorgimento, oltre che scongiurare il rischio di errori di «comando sconosciuto», rende più rapido il caricamento dei programmi stessi in quanto il sistema operativo va a cercarli a colpo sicuro nel posto giusto e non perde tempo girando per tutte le directory del percorso indicato con il comando «path».

Le due righe finali della nostra Startup-Sequence dimostrativa servono al caricamento dell'ambiente WorkBench (con il comando «**LoadWb**») ed alla chiusura della finestra Cli iniziale, per consentire l'accesso all'ambiente ad icone.

L'ultima riga, in particolare, introduce una caratteristica interessante sulla quale è bene soffermarsi: il cosiddetto «**redirezionamento dell'output**».

OUTPUT E REDIREZIONE

La maggior parte dei comandi AmigaDos genera, sullo schermo, una forma di output, ovvero produce qualche messaggio o scritta risultante dalla sua esecuzione.

Se cancellate ad esempio il file «**Pippo**» con il comando «**Delete pip#?**», otterrete come output la scritta «**Pippo deleted**»; allo stesso modo, molti altri comandi generano messaggi di errore o di conferma durante l'esecuzione.

Tutte queste scritte sono utili se il comando viene digitato in modo diretto da prompt; ma se il comando si trova all'interno della Startup-sequence, può essere meglio evitare la loro apparizione, anche per ragioni estetiche.

Per questo motivo si può ricorrere alla redirezione dell'output: si può cioè fare in modo che i messaggi generati da un comando o da un programma finiscano altrove invece che sullo schermo.

Normalmente la redirezione viene usata per inviare su disco o su stampante il risultato di un comando: ad esempio, può essere utile per stampare il contenuto di un file o la directory di un dischetto su carta. Il simbolo di redirezione è l'operatore «>» (maggiore di), che viene usato secondo questa sintassi:

«comando» > «dispositivo o nomefile»

Se ad esempio digitate «**dir >ram:prova**» otterrete come risultato un file di testo di nome «**prova**» nel disco Ram, contenente il listato della directory corrente.

Per stampare il contenuto di un file di testo, si può invece ricorrere al comando «**Type >prt: nomefile**». Come potete notare, la redirezione deve essere usata immediatamente dopo il nome del comando al quale si riferisce: nell'esempio precedente, la sintassi «**Type nomefile >prt:**» avrebbe generato solo un messaggio di errore.

Inoltre, quando la redirezione è riferita ad un dispositivo (**device**), deve essere disponibile anche l'**handler** ad esso relativo: nel caso della stampa, il **device «prt:»** può essere usato solo se il sistema operativo può accedere al «**printer.device**», ovvero se esso è presente nella **directory «devs»** del disco di sistema.

```
AmigaDOS
1> dir df0:
     Trashcan (dir)
     c (dir)
     Prefs (dir)
     System (dir)
     l (dir)
     devs (dir)
*** BREAK
1> dir >ram:Prova df0:
1> type ram:Prova
     Trashcan (dir)
     c (dir)
     Prefs (dir)
     System (dir)
     l (dir)
     devs (dir)
     s (dir)
     t (dir)
     fonts (dir)
     libs (dir)
     Empty (dir)
     Utilities (dir)
     Expansion (dir)
  .info                          Disk.info
  Empty.info                     Expansion.info
  Prefs.info                     Shell
  Shell.info                     System.info
  Trashcan.info                  Utilities.info
1>
```

Un esempio di redirezione dell'output di un comando (in questo caso "dir").

Ancora una volta, per maggiori informazioni su handler e device, vi rimandiamo al fascicolo 24 di AmigaByte.

Due device molto usati nella redirezione sono «**prt:**» e «**nil:**»: il primo corrisponde alla stampante, e serve come abbiamo visto per dirigere su carta la stampa di ciò che normalmente verrebbe inviato allo schermo; il secondo è, al contrario, un device «inesistente». Tutto ciò che viene diretto a «nil:» scompare senza lasciare traccia, venendo inghiottito come da un buco nero: esso serve infatti per eliminare completamente l'output di un comando.

La redirezione può avvenire anche al contrario, per fornire un input ad un comando, invece che per inviarne altrove l'output. In questo caso, si utilizza il simbolo «<» (minore di).

Nella nostra Startup-sequence l'ultima riga redireziona l'output del **comando «endcli»** al **device «nil:»** proprio per evitare l'apparizione sullo schermo del messaggio «**CLI task 1 ending**», che normalmente accompagna l'esecuzione di questo comando.

Un comando importantissimo di AmigaDos, ricorrente in qualsiasi Startup-sequence, è «**run**»: come quasi tutti sanno, esso serve per lanciare altri comandi o programmi in multitasking. La differenza ad esempio tra l'utilizzo delle forme «**dir**» e «**run dir**» risiede nel fatto che nel primo caso il sistema operativo dedica tutta la propria attenzione all'esecuzione del comando, ovvero non consente l'inserimento di altri comandi prima che l'esecuzione di quello in corso sia terminata; nel secondo caso, il prompt riappare subito ed è possibile lanciare un nuovo comando anche mentre quello precedente è ancora in fase di esecuzione.

Il comando «run» è indispensabile per sfruttare adeguatamente le capacità di multitasking di AmigaDos: senza di esso, sarebbe impossibile lanciare contemporaneamente due o più programmi tramite la Startup-sequence, in modo che restino sempre residenti ed in funzione.

Il mancato utilizzo di «run» è la causa principale dei guai di chi procede a personalizzare la Startup-sequence di un proprio dischetto, inserendovi programmi da lanciare dopo il boot del computer.

Facciamo un esempio pratico: supponiamo di possedere un programma che, per funzionare, richiede che la memoria espansa all'interno del computer sia disabilitata mediante l'uso dell'utility «**NoFastMem**». Se si vuole evitare di dover clickare ogni volta sull'icona di «NoFastMem» prima di caricare il programma in questione, si può inserire il lancio di «NoFastMem» nella Startup-sequence del disco in modo che esso entri automaticamente in funzione dopo il boot.

L'utente sprovveduto in questi casi modifica la Startup-sequence, inserendo magari il richiamo a «NoFastMem» prima del caricamento del WorkBench, in questo modo:

nofastmem
loadwb
endcli >nil:

Resettando il computer, si avrà l'amara sorpresa di assistere al blocco totale di Amiga dopo qualche secondo, con la finestra CLI che rimane aperta senza scomparire per lasciare il posto all'ambiente WorkBench.

La ragione è semplice: la Startup-Sequence lancia «NoFastMem», ma non può più proseguire con il comando seguente fino a quando l'esecuzione del precedente non è terminata. E poiché «NoFastMem» è un programma residente, che resta sempre attivo ed in funzione, la Startup-sequence resta perennemente in attesa.

La soluzione è semplice:

...
run <nil: >nil: nofastmem
loadwb
endcli >nil:

In questo modo «NoFastMem» verrà lanciato sfruttando il multitasking e l'esecuzione della Startup-Sequence potrà continuare tranquillamente; la strana sintassi usata per il comando «run», con le due redirezioni al device «nil:», è necessaria per permettere la corretta chiusura della finestra CLI. Infatti una caratteristica particolare di AmigaDos fa sì che una finestra CLI non possa essere chiusa fino a quando sono ancora attivi comandi lanciati da essa: redirezionando input ed output al device «nil:» si può ovviare a questa limitazione.

LA PROVA DEL FUOCO

La nostra Startup-sequence di prova è praticamente conclusa: salvatela quindi premendo il **tasto Esc** e poi il **tasto «X»** seguito da **Return**. Dopo aver atteso lo spegnimento della spia del drive, resettate il computer ed attendete il caricamento del WorkBench che, se non avete commesso errori, dovrebbe apparire come di consueto.

Attivate il **CLI**, impiegando la procedura descritta all'inizio e, quando sarà apparsa la finestra con il prompt, togliete il dischetto del WorkBench modificato, inserendone un altro qualsiasi. Verificate ora la presenza dei comandi AmigaDos residenti, digitando «**dir df0:**»: apparirà la directory del disco presente nel drive interno di Amiga, senza che sia stato necessario reinserire il dischetto WorkBench per caricare il comando «dir».

I possessori di WorkBench 1.3 sappiano comunque che esiste un metodo più immediato per rendere residenti in memoria e direttamente accessibili i comandi AmigaDos: l'uso del comando «**Resident**».

Si tratta però di un comando concettualmente più complesso, del quale ci occuperemo in dettaglio la prossima volta, analizzando un altro argomento «misterioso» di AmigaDos ad esso collegato: i bit di protezione dei file.

□

SUGGERIMENTI E TRUCCHI VARI

I programmatori della BullFrog, già autori del celeberrimo «Popolous», hanno fatto nuovamente centro con «Flood», un simpatico platform game distribuito dalla Electronic Arts.

Quiffy, l'animaletto protagonista del gioco, deve districarsi lungo 42 livelli che vengono progressivamente inondati d'ac-

qua, ripulendoli dalle lattine e dalla spazzatura che li infesta, prima di essere sommerso ed annegare.

Per aiutare Quiffy nella sua impresa, presentiamo tutte le password necessarie per accedere direttamente ad ognuno dei 42 livelli di gioco.

1 FROG	2 YEAR
3 QUIF	4 LONG
5 WORD	6 FRED
7 WINE	8 GRIP
9 TRAP	10 THUD
11 FRAK	12 VINE
13 JUMP	14 NILL
15 FOUR	16 GRIT
17 ZING	18 JING
19 LIDO	20 POOL
21 HATE	22 REED
23 LIME	24 QUID
25 WING	26 FLEE
27 GIGA	28 HEAD
29 LOOP	30 SING
31 JOUX	32 PINK
33 GOGO	34 LETS
35 QUAD	36 BRIL
37 EGGS	38 HENS
39 NAIL	40 SOAP
41 FOAM	42 MEEK

Terminato l'ultimo livello, il gioco finisce tragicamente con una sequenza animata piuttosto macabra: il povero Quiffy infatti emerge da un tombino, solo per essere immediatamente spiaccicato dal pneumatico di un'automobile. Evidentemente ai programmatori della BullFrog piace l'umorismo nero...

Esistono alcune spettacolari funzioni di manipolazione delle immagini in «DemoMaker», il potente programma creatore di demo animate pubblicato sul dischetto allegato al fascicolo 23 di AmigaByte, originariamente non elencate nella documentazione allegata al programma per ragioni di spazio.

Colmiamo ora questa lacuna, rivelandovi che è possibile inserire alcuni comandi particolari per far ruotare e distorcere in vari modi l'immagine visualizzata sullo schermo. Per ottenere questi effetti occorre ricorrere alla pressione di alcuni tasti funzione mentre si digita il testo che dovrà scorrere in fondo allo schermo tramite l'opzione «Edit Text»:

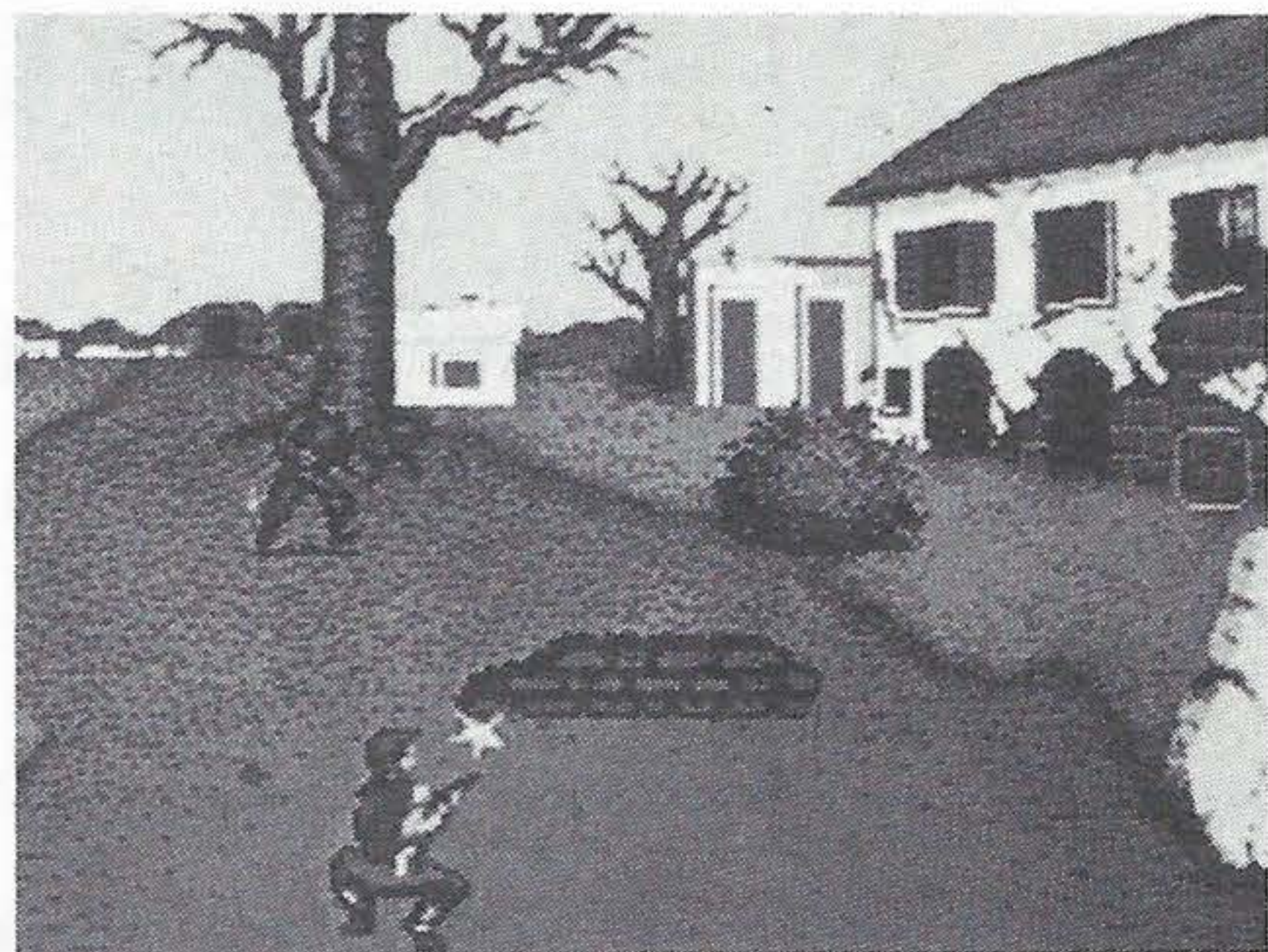
F1 = disattiva ogni effetto.
F2 = effetto «rotazione orizzontale».
F3 = effetto «acqua».
F4 = effetto «distorsione».
F5 = effetto «onda verticale».
F6 = effetto «zig zag».
F10 = cancella il testo presente in memoria.
ALT destro = fa avanzare il testo di 16 caratteri.
AMIGA destro = fa indietreggiare il testo di 16 caratteri.

L'intensità di rotazioni ed effetti si può regolare tramite i gadget sullo schermo.

Il lettore Manuel Giorgini di Venezia, tramite BBS2000 (il Bulletin Board System di AmigaByte), ci ha inviato via modem in redazione un utile suggerimento per la versione 1.1 di «SimCity».

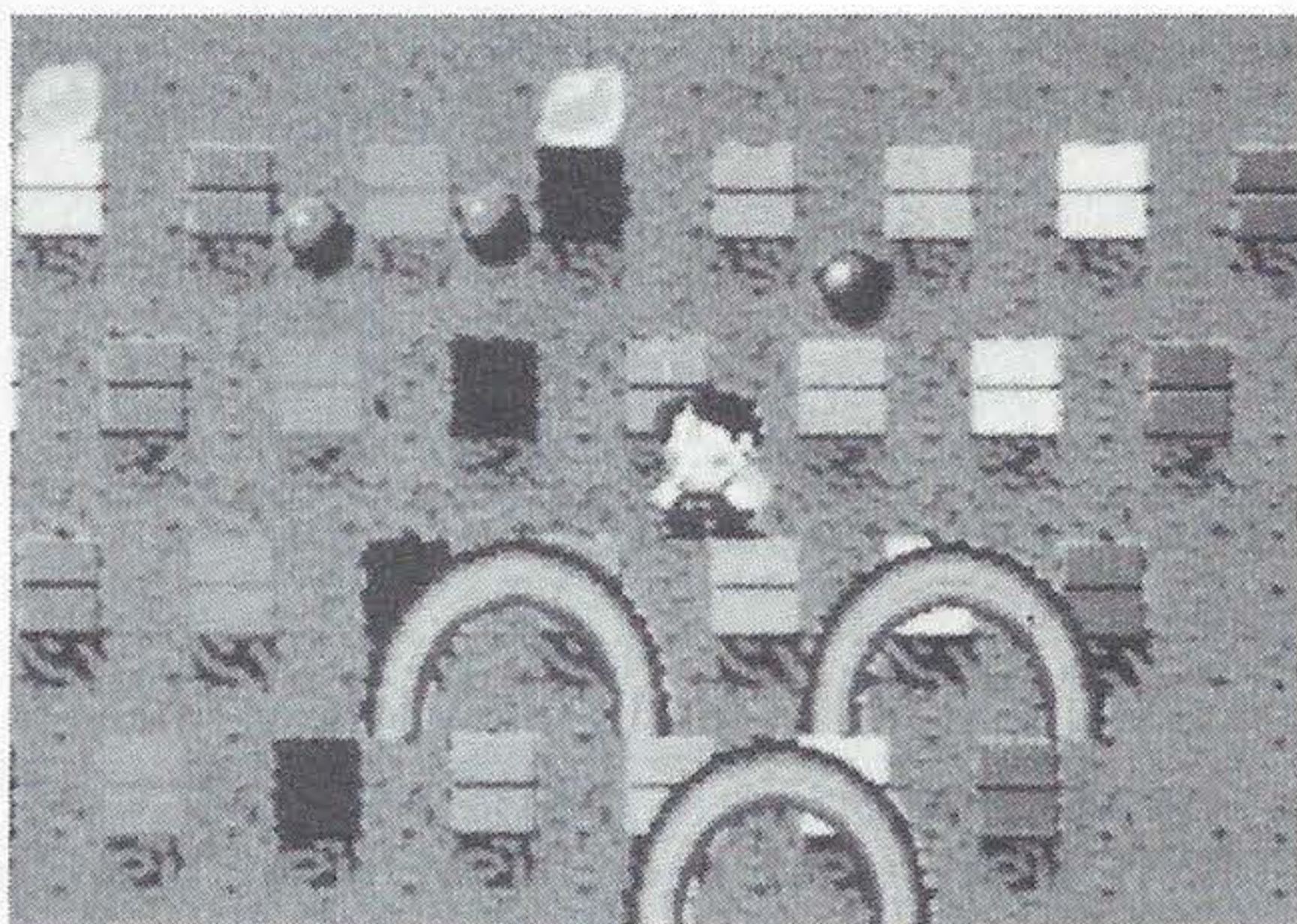
Quando dovete fronteggiare un'inondazione (flooding), è possibile circoscrivere ed eliminare le acque coprendole con del verde, clickando prima sull'icona del parco e poi nei punti desiderati. Dopo aver posizionato il verde, potete usare il bulldozer per spianarlo e l'area tornerà istantaneamente edificabile. Grazie, Manuel!

Alcuni cheat-mode per ottenere vite infinite ed altre diavolerie: il primo è per «Cabal», il gioco di guerra della Ocean. Iniziate a giocare e digitate la parola SCHLIKA sulla tastiera. Se il bordo lampeggia, si-

gnifica che il trucco è stato attivato, permettendovi di premere il tasto F2 per terminare il livello di gioco attuale.

Dai mitragliatori alle arti marziali: in «Shinobi», provate invece a mettere in pausa il gioco ed a digitare LARSXVIII per ottenere crediti infiniti.

In «Strider» invece, provate ad inserire la pausa (con il tasto F9) ed a premere poi contemporaneamente i tasti HELP, SHIFT sinistro, ed 1. Togliete la pausa e potrete premere i tasti 1, 2, 3, 4 e 5 per saltare al livello desiderato, oppure F1, F2, F3 ed F4 per avanzare nell'ambito del livello attuale.

In «Rainbow Islands», il seguito del celebre gioco arcade «Bubble Bubble», esiste una stanza segreta piena di pozioni magiche. Per fare apparire nella tana del ragno-guardiano la porta argentea che conduce alla stanza in questione, occorre raccoglie-

re i diamanti seguendo un ordine ben preciso: quello dei colori dell'arcobaleno.

La sequenza è quindi la seguente: rosso, arancione, giallo, verde, azzurro, indaco, violetto.

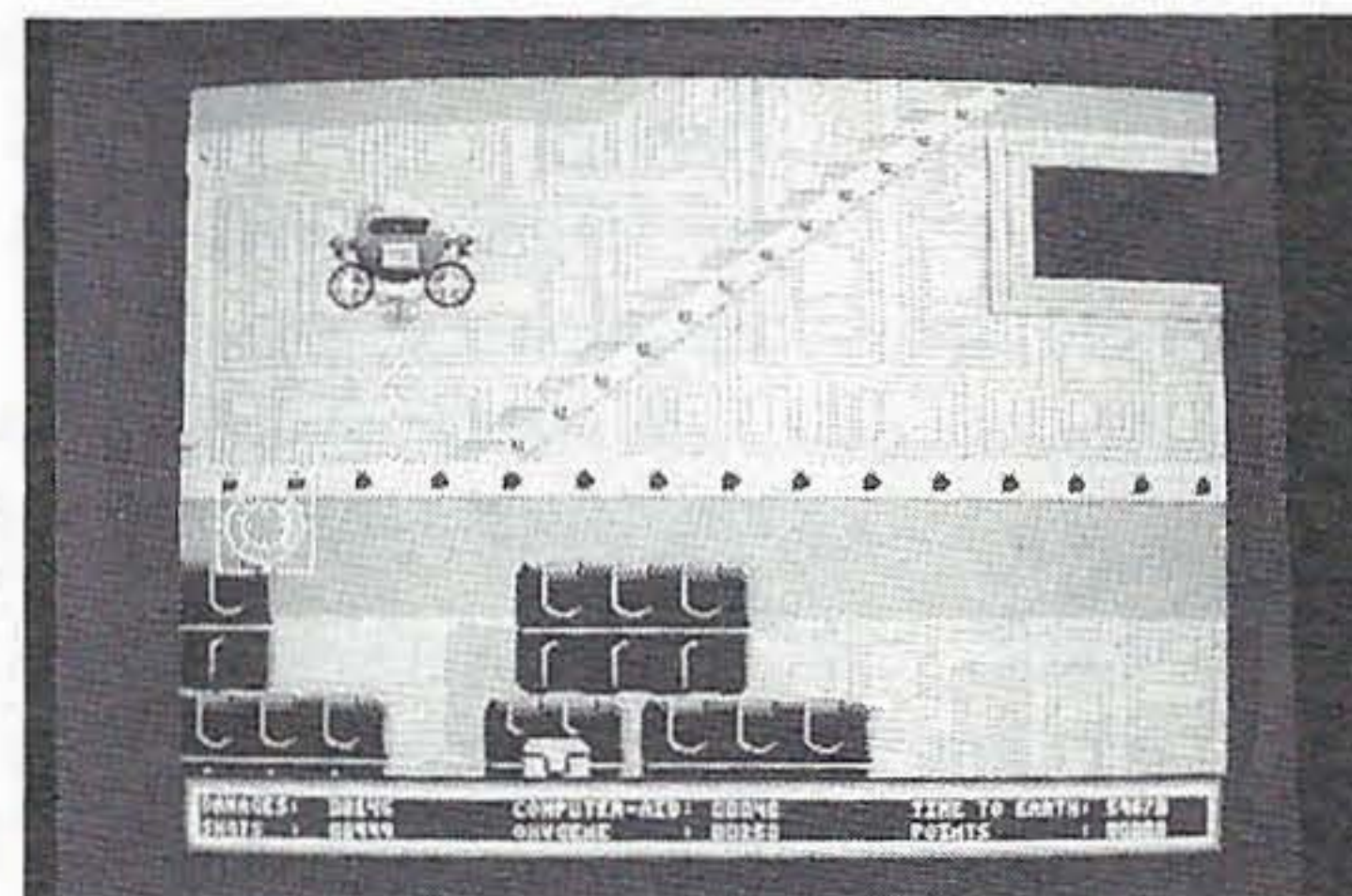
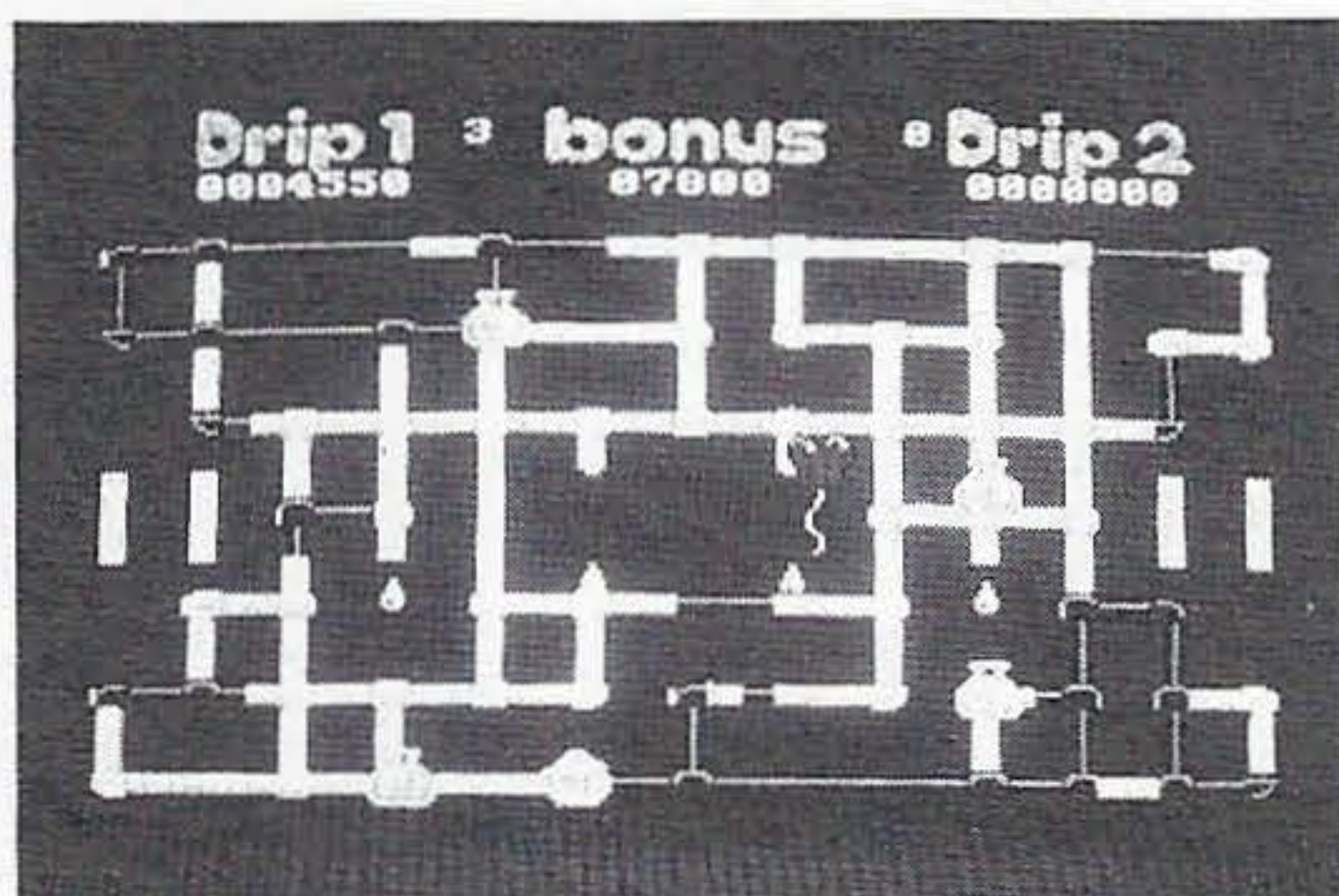
WANTED

SEI TU IL PROSSIMO ABBONATO AD AMIGA BYTE

Per 11 fascicoli ed altrettanti dischetti direttamente a casa tua

IN PIÙ IN REGALO UN SUPERDISCO

CON DUE SPLENDIDI GIOCHI INEDITI

** Il superdisco viene inviato anche a chi si abbona a prezzo ridotto (L. 85mila) per ricevere esclusivamente i fascicoli senza dischetto allegato.*

ABBONATI! Cosa aspetti?

Per abbonarti invia vaglia postale ordinario ad Arcadia srl, c.so Vitt. Emanuele 15, 20122 Milano.

Real 3D, la potenza della velocità

Simula qualsiasi tipo di superficie, supporta le macro, ha proprio tutto ciò che serve ed anche qualcosa d'altro. In più, è terribilmente veloce.

di LUCA MIRABELLI

Fra tutti i procedimenti utilizzati in computer graphics, il **ray-tracing** detiene senz'altro due primati: quello della spettacolarità dei risultati, e quello meno gradevole della lunghezza dei tempi di realizzazione.

Questa tecnica di rappresentazione consiste nel ricostruire la scena vista da un ipotetico osservatore partendo da una descrizione matematica degli oggetti in essa presenti, e calcolando il percorso di tutti i raggi di luce che, partendo da una fonte luminosa, giungano al punto di osservazione.

Bello, vero? Purtroppo un simile procedimento richiederebbe svariati miliardi di calcoli, e non sarebbe completabile in tempi umani.

Dovremmo dunque rinunciare in partenza a questo potentissimo strumento? La risposta è ovviamente no, e l'esistenza stessa di programmi come «Sculpt-4D», «Turbo Silver» e «**Real 3D**», del quale parleremo tra un attimo, lo prova. La scappatoia è offerta dalle applicazioni euristiche.

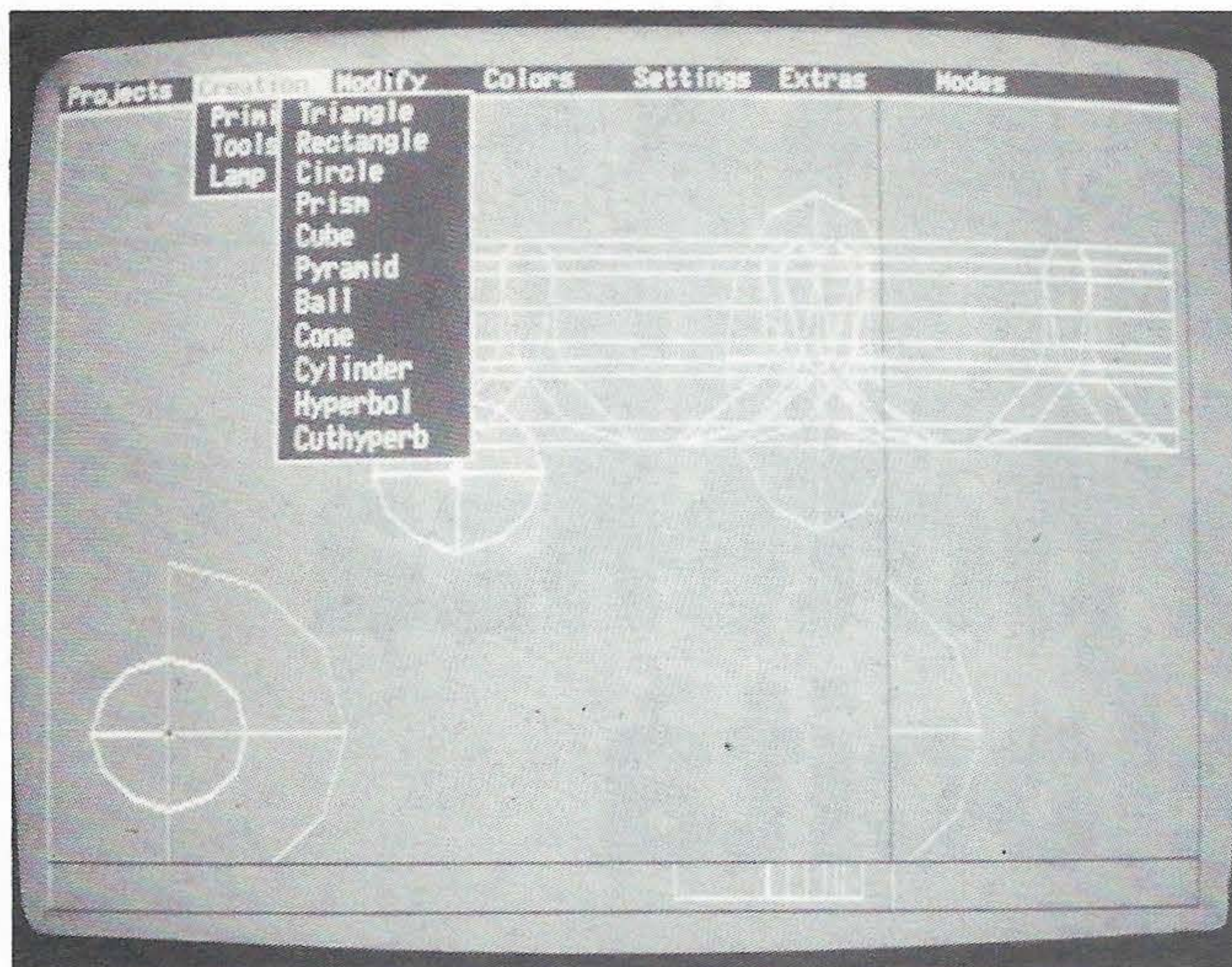
L'EURISTICA COS'È

L'euristica è una branca dell'Intelligenza Artificiale che si occupa di elaborare algoritmi che, a prezzo di un piccolo abbassamento della qualità del risultato, consentono grandi economie in termini di complessità di calcolo e di tempi di elaborazione. Nel caso specifico del ray-tracing, la realizzazione di un'immagine può essere velocizzata di centinaia e centinaia di volte.

Un secondo handicap di questa tecnica è la difficoltà di descrivere le forme del mondo che ci circonda in termini matematici, cioè suddividendole in coni, cubi, rettangoli, ed in altre forme geometriche primitive. Per venire incontro all'utente, i migliori programmi consentono di disegnare oggetti a mano libera, e si occupano della scomposizione in modo del tutto automatico.

VECCHIO SCULPT, ADDIO PER SEMPRE?

Ma veniamo all'oggetto principale del nostro interesse, il programma «**Real 3D**». I programmatori hanno implementato tutte le funzioni umanamente desiderabili dall'utente, sia medio che evoluto, ed il risultato è stato davvero notevole. Caricato il programma, sullo schermo appaiono quattro finestre. Le tre maggiori costituiscono i piani di una proiezione ortogonale, il nostro ambiente di lavoro per quanto riguarda la progettazione della scena. La quarta finestra è relativa alla selezione degli oggetti. «Real 3D» organizza gli

Fig. 1: Questa immagine è stata generata da Real 3D in circa due ore.

Così si presenta l'ambiente di lavoro nella fase di progettazione

oggetti di una scena secondo una struttura gerarchica: l'oggetto «bullone», ad esempio, è formato da un cilindro e da un prisma a base esagonale. Poiché quest'ultimo non è presente tra le primitive offerte dal programma, è stato realizzato con sei parallelepipedi angolati di sessanta gradi. Effettuare una qualsiasi operazione su «bullone» oppure singolarmente su tutti i suoi componenti, è la stessa cosa, ma è evidente la praticità del primo metodo.

La finestra di selezione degli oggetti riporta, nella parte alta, il nome di un oggetto e, nella parte bassa, l'elenco dei suoi componenti, che possono essere a loro volta oggetti oppure primitive geometriche. E così via, all'infinito... Ci si può muovere agevolmente su e giù per questa gerarchia a colpi di mouse: un click sul titolo della finestra ci porterà al livello superiore, mentre uno sul nome di un oggetto ci illustrerà in dettaglio i suoi componenti.

Vediamo ora come sia possibile aggiungere livelli o componenti alla gerarchia. Con **Object Create** (primo menu) creiamo un sottogruppo di quello in cui ci troviamo attualmente, mentre **Object Create root** serve ad aggiungere un livello al di sopra di esso.

Il secondo menu provvede ad aggiungere nuovi componenti al gruppo attualmente selezionato. Questi possono essere primitive geometriche (primo sottomenu), o altri oggetti. In quest'ultimo caso, abbiamo tre strumenti a nostra disposizione per crearli:

- **Lathe** (= tornio), per creare solidi di rotazione (saranno scomposti in coni e tronchi di cono);
- **Circular Tube**, un tubo a sezione circolare che possiamo piegare a nostro piacimento (verrà scomposto in cilindri e sfere):
- **Rectangular Tube**, come sopra ma a sezione rettangolare (dà origine a parallelepipedi raccordati da sfere).

Fig. 2: Differenti tipi di avvolgimento: Spiral a sinistra, Ball a destra.

OPERAZIONI LOGICHE FRA OGGETTI

Le possibilità fin qui descritte sono già notevoli, ma non abbiamo ancora affrontato una caratteristica unica di «Real 3D»: le operazioni logiche tra oggetti. Presenti nel terzo menu, esse sono AND, AND NOT, EOR e DIVIDE; per eseguirle occorre selezionare un oggetto, poi l'operazione desiderata, quindi un secondo oggetto, nel quale sarà memorizzato il risultato. Il primo oggetto potrà quindi essere tranquillamente cancellato.

- **AND** equivale ad un'intersezione: faranno parte del risultato tutti e solamente i punti che erano condivisi tra i due oggetti di partenza. Ad esempio, un AND tra due sfere compenetrate darà origine ad una forma biconvessa, mentre un AND tra due solidi separati avrà risultato nullo.
- **AND NOT** genera un nuovo solido formato da tutti i punti che facevano parte del secondo ma non del primo; è come se si scavasse un buco nel secondo oggetto, utilizzando il primo come punta di trapano. Il dado di figura 1 è il risultato di un AND NOT tra un cilindro ed un prisma a base esagonale.
- **EOR** genera un solido equivalente all'intersezione dei due di partenza, ma con l'esclusione della zona in comune. Per ovvie ragioni, il risultato di questa operazione è visibile solamente se gli oggetti non sono totalmente opachi.
- Per concludere, **DIVIDE** spezzerà il secondo oggetto in due parti, che conterranno rispettivamente i risultati di un AND e di un AND NOT, e saranno chiamate oggetto_1 ed oggetto_2.

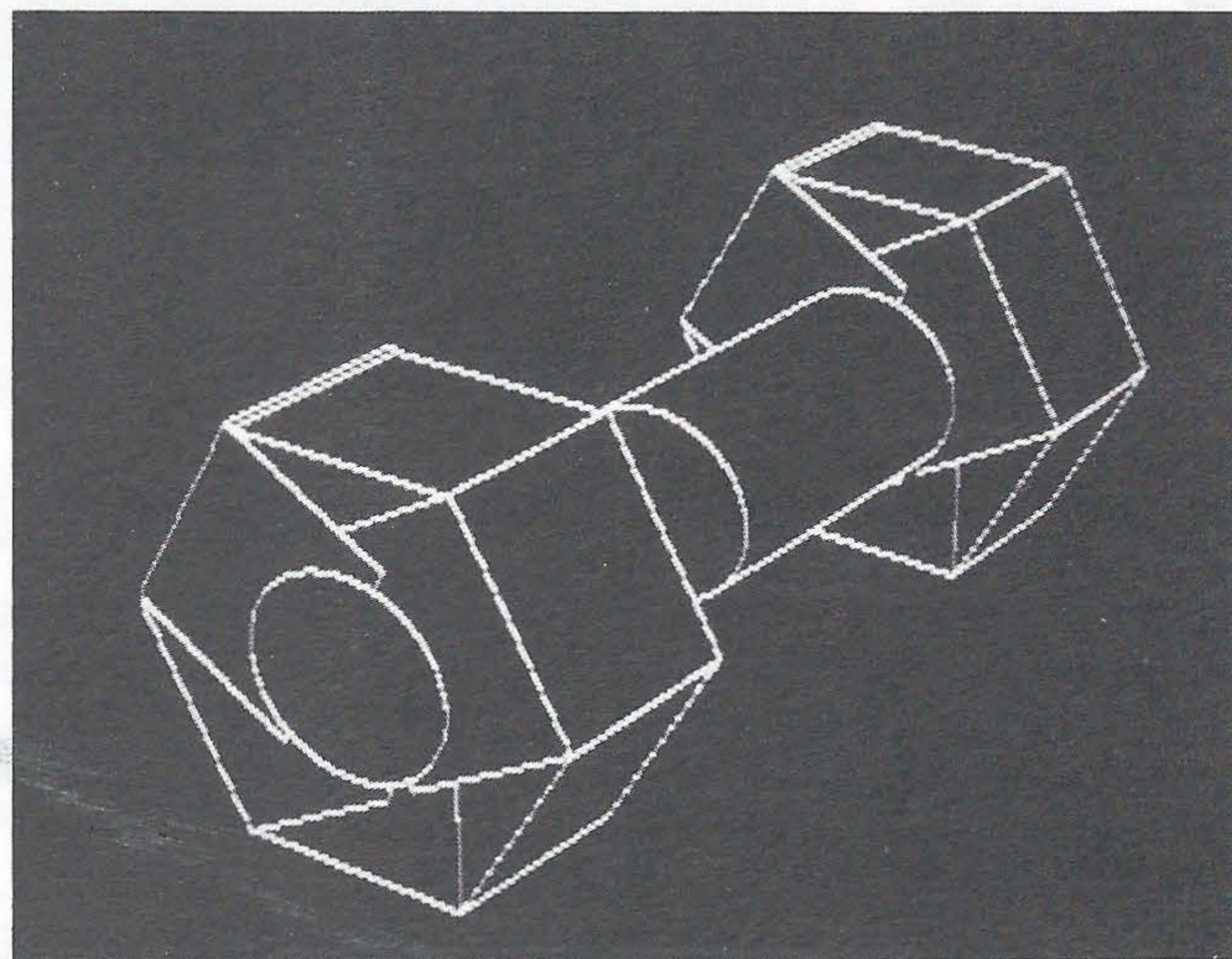



Fig. 3: Il modo Outline evidenzia i contorni tra i singoli oggetti.

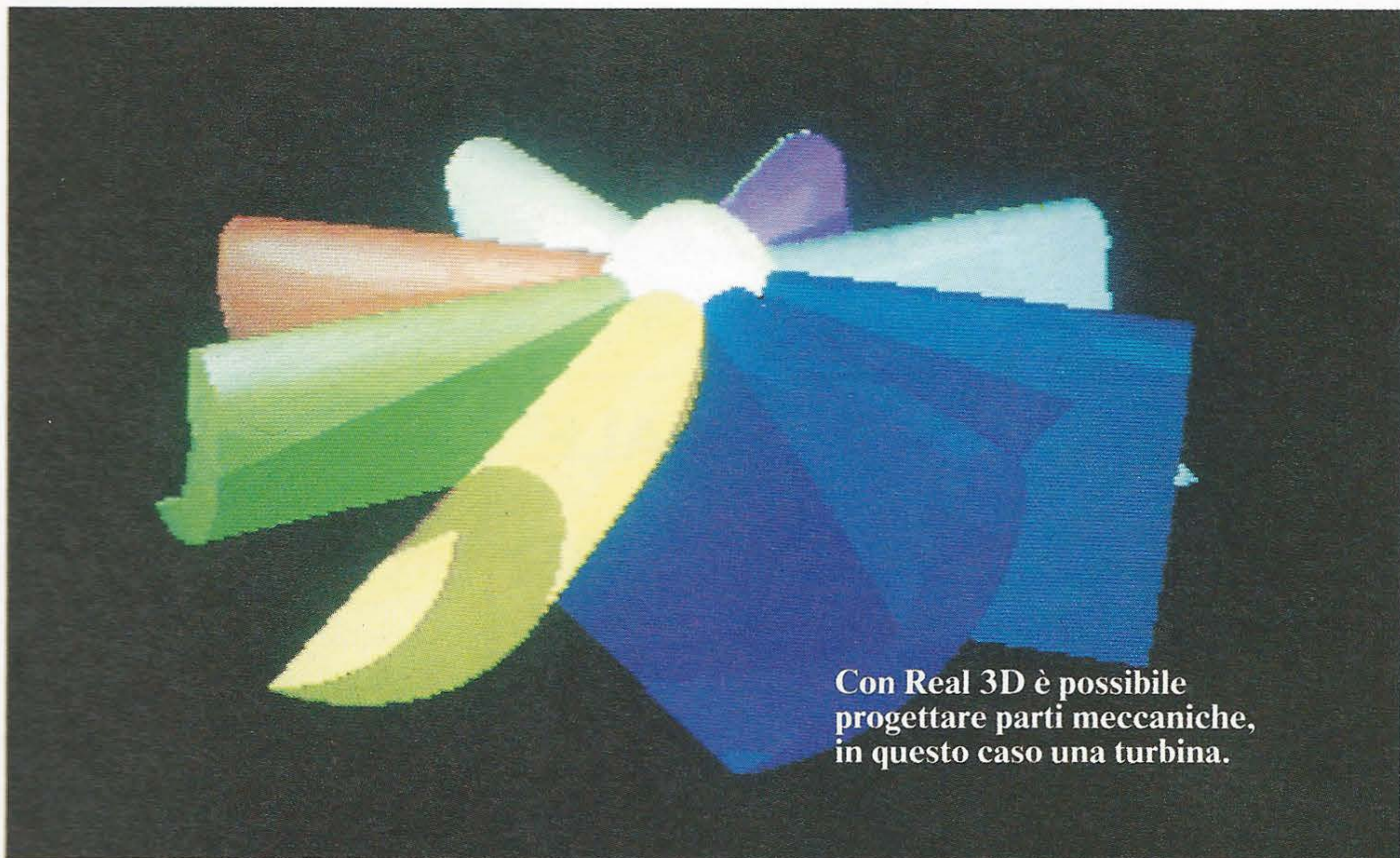
Le operazioni logiche sono un aiuto davvero fondamentale in fase di progettazione, ed è necessario imparare ad utilizzarle ed a prevederne le conseguenze, se si vuole utilizzare il programma a livello professionale.

COME TI TRASFORMO L'OGGETTO

Il terzo menu, sotto la

voce **Hierarchy**, contiene tutte le trasformazioni più o meno «cruente» alle quali è possibile sottoporre un oggetto. Sono ben quindici, e vale la pena di conoscere anch'esse a fondo.

- **Move** muove un oggetto in una certa direzione, memorizzando lo spostamento relativo;
- **Move To** ha la stessa funzione, ma memorizza le coordinate del punto di arrivo (la differenza tra queste due funzioni apparirà più evidente quando affronteremo le macro);
- **Size** cambia le dimensioni di un oggetto senza alterarne le proporzioni;
- **Stretch** cambia sia le dimensioni che le proporzioni di un oggetto;
- **Extend** modifica una dimensione a scelta dell'oggetto. Tale dimensione può anche non essere parallela agli assi della proiezione;
- **Rotate** fa ruotare un oggetto attorno ad un punto su uno dei tre piani di proiezione (rotazioni su altri piani devono essere scomposte nelle loro componenti ortogonali);
- **Mirror** ribalta un oggetto rispetto ad un piano perpendicolare ad uno dei tre di proiezione;
- **Explode** agisce soltanto sui gruppi, allontanandone fra loro i componenti;
- **Copy** crea una copia dell'oggetto attualmente selezionato;
- **Rename** cambia il nome ad un oggetto;
- **Locate** ne cambia la posizione all'interno della gerarchia;
- **Delete** lo elimina (attenzione! dopo la conferma non sono possibili ripensamenti, e l'oggetto sarà perso per sempre)
- **Color** consente di attribuire il colore attualmente selezionato (menu **Colors**) ad un oggetto;

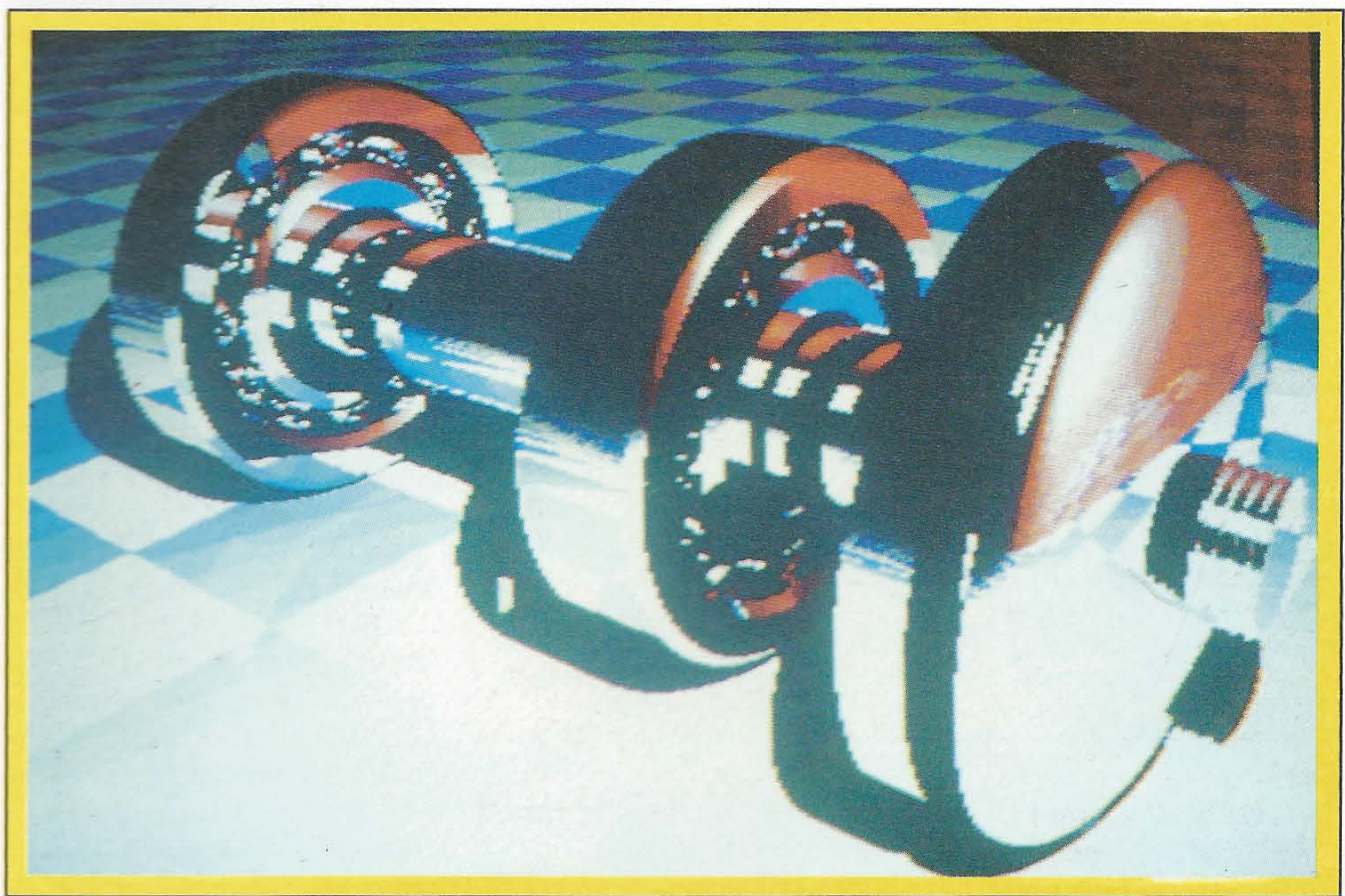
Prima di descrivere le ultime due funzioni, vale la pena di trattare brevemente di un'altra caratteristica di «Real 3D», questa volta non unica, ma insolitamente potente.

Con Real 3D è possibile progettare parti meccaniche, in questo caso una turbina.

LA GESTIONE DEI MATERIALI

Un materiale è definito da tre parametri fisici: lucentezza (**brilliance**), trasparenza (**transparency**), ed indice di rifrazione (**speed of light**). Inoltre, ad ogni materiale può essere associata un'immagine,- che verrà utilizzata per «avvolgere» la superficie degli oggetti. Per quest'ultima operazione, detta **Painting**, sono disponibili quattro modalità:

- **Parallel** proietta perpendicolarmente il disegno sulla superficie. Se questa è curva, il disegno verrà deformato;
- **Cylinder** proietta il disegno perpendicolarmente in altezza, ma lo «piega» nel senso della larghezza, per assecondare la curvatura dell'oggetto;
- **Ball** adatta il disegno ad una superficie sferica con la minima deformazione possibile;
- **Spiral** avvolge il disegno a spirale. In figura 2 si possono cogliere le differenze fra le modalità Spiral e Ball.

I comandi per modificare, creare, salvare e caricare materiali si trovano nel primo menu.

Dopo aver esaminato le caratteristiche della gestione dei materiali in «Real 3D», è facile comprendere le ultime due voci del sottomenu Hierarchy:

- **Material** attribuisce un materiale all'oggetto attualmente selezionato;
- **Painting** definisce la direzione secondo cui il disegno verrà avvolto attorno all'oggetto. Di default, se l'oggetto presenta degli spigoli, la direzione di avvolgimento è parallela ad uno di essi.

Gli strumenti di progettazione inclusi in «Real 3D» sono decisamente sufficienti per realizzare qualsiasi idea, e sicuramente superiori a quelli di tutti gli altri concorrenti; ciononostante, mantengono una notevole facilità ed immediatezza di utilizzo.

Terminata la fase di definizione degli oggetti, occorre inserire altri importanti elementi nella scena: le fonti di luce. L'equivalente software del biblico «fiat lux» si trova nel secondo menu e risponde al nome di **Lamp**. La sorgente luminosa può subire tutte le trasformazioni finora esaminate: alcune di esse (explode, stretch, extend, rotate, solo per fare alcuni esempi) non hanno però alcun effetto sull'immagine definitiva. Le più utili, invece, sono **Size** (modifica le dimensioni, quindi l'intensità della lampadina) e **Color** (non dimentichiamo che anche la luce ha un suo preciso colore).

DEFINIRE UN MATERIALE

I parametri necessari a «Real 3D» per identificare un materiale sono tre (Brilliancy, Transparency, e Speed of Light): esaminiamo ora il loro significato pratico, per essere in grado di simulare tutto ciò che ci sta intorno.
I primi due non dovrebbero offrire alcuna difficoltà: chiunque è in grado di stabilire quanto una superficie sia lucida, e quanto sia trasparente: uno specchio avrà trasparenza zero e lucidità massima, mentre con lucidità e trasparenza medie otterremo una superficie traslucida, come il vetro smerigliato.
L'indice di rifrazione ha a che fare con le deviazioni che i raggi di luce subiscono nel passaggio da un materiale ad un altro, a causa della differente velocità della luce nei due mezzi.
Tipiche applicazioni della rifrazione sono le lenti (in «Real 3D» una lente biconvessa è facilmente ottenibile come risultato dell'AND tra due sfere di largo raggio).
La velocità della luce in un materiale è inversamente proporzionale alla sua densità: il cristallo, che ha peso specifico più elevato del vetro a causa della presenza di ossidi di piombo, è caratterizzato da una velocità di propagazione della luce assai minore.
Nel caso di solidi con trasparenza nulla, poi, la velocità della luce è assolutamente irrilevante, ed il programma non ne terrà conto.

Le due sfere hanno alto indice di rifrazione, quindi bassa velocità della luce.

DA UN CERTO PUNTO DI VISTA...

Non meno importante è comunicare al programma dove siamo ed in quale direzione stiamo guardando. Gli appositi comandi si trovano nel sesto menu, intitolato **Extras**. L'opzione **Observer** ci permette di specificare la posizione del punto di vista (simboleggiato da un cerchietto), mentre per mezzo dell'opzione **Look** potremo specificare il punto verso il quale stiamo guardando. Questo «mirino» sarà rappresentato da un asterisco.

Il passo successivo è l'esame preliminare della scena in modo **Wireframe**, vale a dire nel modo in cui sono disegnati soltanto i contorni degli oggetti: per accedervi, selezioniamo l'opzione omonima dell'ultimo menu. Comparirà l'immagine tanto desiderata della nostra scena e, nella parte bassa dello schermo, un pannello ci permetterà di muovere il punto di vista.

Il piccolo quadratino al centro del rettangolo più grande va usato come un joystick: tempestandolo di click ci muoveremo nella direzione specificata, mantenendo costante la distanza dal «mirino»: in pratica, ci sposteremo lungo i meridiani ed i paralleli di una superficie sferica.

Il primo dei due cursori sulla destra serve a modificare la distanza dal mirino, il secondo amplia o riduce il nostro campo di osservazione; è preferibile evitare di spostarlo completamente a sinistra, a meno che non si desiderino gli effetti di deformazione tipici degli obiettivi grandangolari.

Terminate le correzioni, bisogna ricordarsi di selezionare il pulsante contrassegnato dalla scritta **REC** per memorizzarle, altrimenti andranno perse. L'utente medio di «Real 3D» va incontro a crisi depressive ricorrenti, poiché dimenticare questa piccola operazione obbliga a cominciare da capo il lavoro di rifinitura.

Se invece tutto è andato come doveva, siamo pronti per passare alla fase di ray-tracing vera e propria, cosa che faremo immediatamente con l'opzione **Volume** dell'ultimo menu: apparirà un pannello per mezzo del quale comunicheremo al programma le modalità ed i parametri del procedimento di ray-tracing.

LA GENERAZIONE DELL'IMMAGINE

Tanto per cominciare, nella parte alta troviamo il nome dell'immagine (che inizialmente corrisponde al nome del livello più alto della gerarchia di oggetti): al termine dell'elaborazione, il risultato sarà salvato su disco (nella directory **REAL:Pictures**, se il nome non comprende un path differente).

I primi tre cursori hanno due funzioni differenti, a seconda che il pulsante alla loro sinistra si trovi in modalità **Background** oppure **Baselight**. Nel primo caso specificheranno le componenti RGB del colore di fondo, nel secondo caso il colore della luce diffusa.

Salvo esigenze particolari, è opportuno lasciare lo sfondo nero e la luce diffusa ad un valore attorno alla metà del massimo per tutte e tre le componenti: servirà ad evitare che le parti non direttamente illuminate degli oggetti risultino troppo scure o addirittura nere.

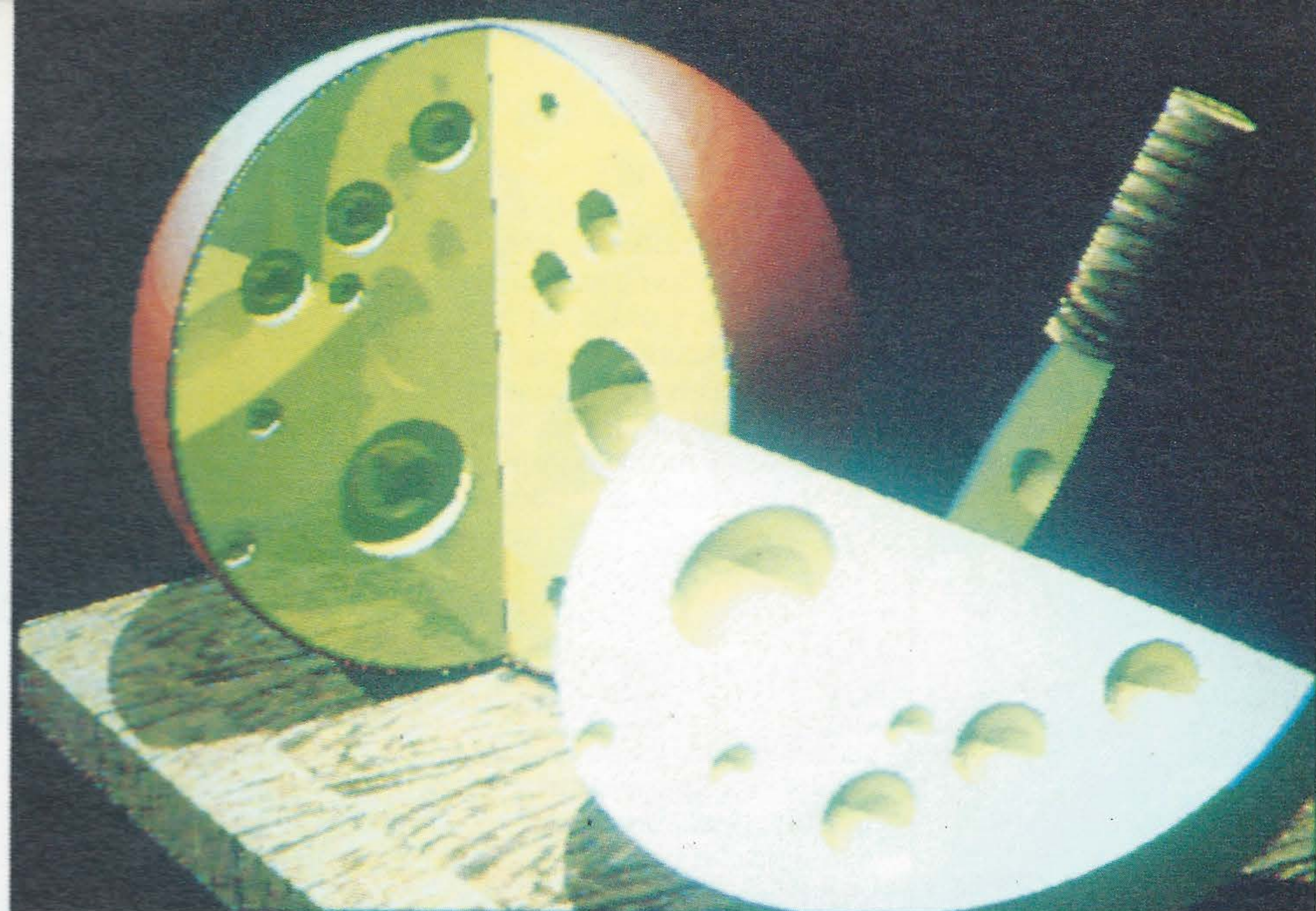
Il cursore denominato **Brightness** definisce la luminosità globale della scena (cioè l'intensità delle fonti di luce che abbiamo definito); viene disattivato selezionando il pulsante **Autolight**. In questo caso il computer dosa automaticamente la quantità di luce presente nella scena, ma attenzione: se le sorgenti sono abbastanza vicine agli oggetti, il nostro caro «elettrodomestico» sbaglierà a valutarne la luminosità, ed il risultato sarà un'immagine decisamente scura. Per questo motivo l'opzione Autolight non è sempre consigliabile.

COME SEMPLIFICARSI L'ESISTENZA

Il pulsante con la scritta **Normal** può disattivare alcune caratteristiche del procedimento di ray-tracing, garantendo aumenti molto vistosi della velocità di esecuzione. Se viene selezionato, il modo di rappresentazione della scena diventa, nell'ordine:

- **Outline**: saranno tracciati soltanto gli spigoli effettivamente visibili, con un risultato simile a quello di figura 3. È l'unica modalità che non lavora in Hold and Modify, ma in Hi-Res, con 640 punti in orizzontale;

- **Fast**: il programma non terrà conto né dei materiali né delle fonti luminose da noi definite, supponendo che ce ne sia una unica coincidente con l'osservatore. Inoltre le ombre non vengono calcolate;

- **Shadowless**: disattiva il calcolo delle ombre portate, per cui l'illuminazione di ciascun oggetto dipenderà solamente dalla sua collocazione rispetto alle fonti luminose, e non da altri oggetti interposti.

Tra il modo Outline ed il modo Wireframe visto in precedenza, vi sono due importanti differenze. La prima è di ordine teorico: il Wireframe è costruito geometricamente, l'Outline otticamente. La seconda è di ordine pratico: un oggetto derivante da un'operazione logica sarà disegnato, in modo Wireframe (così come in fase di progettazione), come l'unione dei due oggetti da cui deriva; soltanto il modo Outline permetterà di ottenere una rappresentazione corretta.

Tornando al pannello, il parametro **Recursion Depth** determina il numero massimo di segmenti del quale sarà tenuto conto nella ricostruzione del cammino dei raggi luminosi. Se in una scena abbiamo un solo oggetto ed una fonte luminosa, questo parametro potrà essere fissato a 2 (il cammino della luce si compirà infatti dalla sorgente all'oggetto, e dall'oggetto all'osservatore). Se avessimo invece due oggetti di cui almeno uno riflettente, sarebbe necessaria una Recursion Depth pari almeno a 3 (dalla sorgente ad un oggetto, poi all'altro, poi al nostro occhio), e così via.

I due parametri **Resolution** stabiliscono la griglia di definizione dell'immagine: da **1*1** (la stessa risoluzione dello schermo) fino a **8*8** (un numero di punti, e di calcoli, sessantaquattro volte minore). Questa caratteristica si rivela molto utile per effettuare un rapido preview (anteprima) del nostro lavoro, ed essere sicuri che tutto funzionerà a dovere quando decideremo di calcolare la scena con più cura.

UNA QUALITÀ PROFESSIONALE

Non finisce qui: per mezzo di appositi pulsanti si possono attivare i modi **Overscan** e **Interlace**, così da portare la risoluzione di

schermo fino ad un rispettabilissimo 352*564. Siete pronti? Selezionate **OK**, e il processo avrà inizio. La durata può variare da meno di un secondo (per un cubo in modo fast con risoluzione 8*8) ad 8 ore e più (per scene molto complesse in overscan interlacciato).

E LE ANIMAZIONI?

I fanatici di «Sculpt» e «Silver» a questo punto staranno commentando ironicamente: «Già, e le animazioni?». Tranquilli, «Real 3D» le permette, anche se all'inizio non è troppo facile abituarsi alla sua filosofia di lavoro.

Il programma vede le animazioni come gruppi di immagini, e come tali le salva su disco: è compito di una utility aggiuntiva, inclusa nel pacchetto e denominata **DeltaConvert**, compattarle in file simili ai più noti file Anim (con i quali tuttavia non saranno compatibili: visualizzare le animazioni è quindi compito esclusivo dell'utility **DeltaPlay**).

Per comunicare al programma che ci accingiamo a lavorare su di un'animazione, occorre specificarne il numero di fotogrammi con il comando **Animation Size** (primo menu). Appariranno allora in alto a destra tre nuovi pulsanti: freccia a sinistra, freccia a destra, ed uno strano simbolo costituito da una X maiuscola tra parentesi. Con le frecce ci si muove avanti ed indietro nell'animazione, mentrre il pulsante centrale serve a contrassegnare i fotogrammi «esposti», vale a dire quelli che presentano delle differenze rispetto a quelli che li precedono. Inizialmente solo il fotogramma 0 è esposto, e qualsiasi modifica effettuata su qualunque fotogramma avrà effetto sull'intera animazione. Se si desidera, pertanto, che un oggetto raddoppi le sue dimensioni nel sesto fotogramma, quest'ultimo deve essere esposto, prima di procedere con il comando Size.

AUTOMAZIONE SEMPRE PIÙ SPINTA

Le macro sono sequenze di istruzioni che i programmi più potenti permettono di associare ad un comando unico. «Real 3D» ne prevede una sola, ma è comunque possibile crearsi una libreria di macro perché esse possono venire salvate e caricate.

Macro Define e **Macro End** (primo menu) contrassegnano l'inizio e la fine della definizione della macro; **Execute** la esegue una singola volta, **Repeat** più volte di seguito.

Ma il comando più interessante per quanto riguarda le animazioni è senza dubbio **Macro Accumulate**.

Questa funzione esegue la macro sul fotogramma attualmente selezionato, passa al successivo, lo espone, ed esegue la macro nuovamente, e così via fino a raggiungere il numero di ripetizioni indicato dall'utente.

Per far compiere ad un oggetto un giro completo su se stesso in 24 fotogrammi, ad esempio, il modo migliore è definire una macro che lo faccia ruotare di 15 gradi (360 diviso 24), portarsi sul primo fotogramma, ed impartire il comando Macro Accumulate con un numero di ripetizioni pari a 23.

MODI E MODE

Appare quindi chiaro che, per muovere un oggetto, occorre creare una macro che sfrutti il comando Move e non il Move To, poiché il primo può essere ripetuto, il secondo no (o meglio, le ripetizioni successive alla prima non avrebbero alcun effetto).

Passando in **modo Wireframe**, è possibile attribuire posizioni differenti del punto di vista per ogni fotogramma, anche per quelli non esposti. Attivando il pulsante con la scritta **AR** (Auto Rec), si passa al fotogramma successivo dopo ogni modifica: è utile per muovere gradualmente il punto di vista.

Per finire, in **modo Volume**, selezionando **OK** verranno generati e salvati i fotogrammi **nome0**, **nome1**, **nome2... nomeN**, dove il «nome» è quello da noi specificato nella parte alta del pannello. Il pulsante **Single**, se attivato, arresterà il processo dopo il primo fotogramma; possiamo inoltre partire da un fotogramma diverso dallo 0 per mezzo dei due pulsanti a freccia in alto.

CONCLUSIONI &... MEDITAZIONI

Tirando le somme, si può affermare che «**Real 3D**» è un programma potentissimo, veloce (provare per credere) ed abbastanza affidabile. Abbiamo rilevato la presenza di un bug, che dà origine ad un errore interno se si vogliono visualizzare oggetti compenetrati trasparenti; si tratta comunque di un errore che il programma è in grado di gestire senza mandare il sistema in tilt.

I programmi di ray-tracing utilizzano solitamente una notevole quantità di memoria: se si attiva l'opzione **Savemem** nel pannello Volume, e soprattutto se non si pretende di visualizzare il David di Michelangelo in overscan interlacciato, una dotazione di un Megabyte è sufficiente per avere da «Real 3D» grandi soddisfazioni. □